Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Aplicación destinada a persoas con discapacidades motrices para o control de elementos do seu entorno

Estudante: Carlos Varela Manteiga
Dirección: Óscar Fresnedo Arias
Francisco Laport López

A Coruña, setembro de 2020.

A meus pais

Agradecementos

En primeiro lugar gustaríame agradecer aos meus directores de proxecto, Óscar Fresnedo Arias e Francisco Laport López, a súa axuda, dedicación e paciencia.

Tamén quero dar as gracias a meus pais, os cales fixeron un gran esforzo para que eu puidese chegar a este punto e sempre me apoiaron e, finalmente, gustaríame dar as gracias a todos os amigos e compañeiros que coñecín ao longo da carreira, xa que gracias a eles vou a recordar esta etapa universitaria con gran cariño.

Resumo

A día de hoxe a utilización de interfaces cerebro-ordenador está propiciando grandes avances en diferentes campos como a medicina, o mundo do entretemento, etc. Este tipo de ferramentas proporcionan ás persoas unha gran variedade de posibilidades, que poden ir dende a súa utilización para o control de videoxogos ata facer máis fácil a vida das persoas con problemas de saúde ou movilidade. Neste proxecto, imos centrarnos nesta última posibilidade, intentando crear unha aplicación que facilite a vida a persoas con problemas severos de movilidade á hora de controlar diferentes dispositivos do seu fogar que, debido ós seus problemas motrices, poden resultar difíciles de usar.

Esta aplicación permitirá aos usuarios que a utilicen controlar diferentes dispositivos que están presentes no día a día de calquera persoa, como poden ser unha televisión, un teléfono móbil ou un termostato, tan só mediante os seus pestanexos. Para iso, vanse implementar os diferentes módulos software necesarios para o seu correcto funcionamento, entre os que se encontrarán os detectores de pestanexos e obxetos ou o módulo de conectividade necesario para a comunicación de todos os dispositivos involucrados na contorna que se vai crear.

Abstract

Nowadays the use of brain-computer interfaces is leading to great advances in different fields such as medicine, the world of entertainment, etc. These types of tools provide people with a wide variety of possibilities, which can range from their use to control video games to making life easier for people with health or mobility problems. In this project we are going to focus on this last posibility, trying to create an application that makes life easier for people with severe mobility problems by controling different devices that, due to the handicap suffered by these people, can become difficult to use.

This application will allow users who use it to control different devices that are present in any person´s day-to-day life, such as a television, a mobile phone or a thermostat, just with their blinkings. For doing this, the different software modules that will be needed for its correct operation will be implemented, among which will be the blinkings and objects detectors or the connectivity module necessary to connect all the devices involved in the environment that is going to be created.

Palabras chave:

- OpenCV
- YOLOv3
- Detector
- BCI
- EEG
- Pestanexos
- MQTT

Keywords:

- OpenCV
- YOLOv3
- Detector
- BCI
- EEG
- Blinkings
- MQTT

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

1.1 Introdución

As interfaces cerebro-ordenador (BCI, do inglés *Brain-Computer Interface*) gañaron unha gran importancia durante os últimos anos, xa que se presentan como unha ferramenta de moita utilidade para pacientes con problemas severos de mobilidade. Estas permiten a un suxeito a interacción con dispositivos da súa contorna facendo uso únicamente dos seus sinais cerebrais, sen necesidade de grandes (ou incluso ningún) movementos musculares. A día de hoxe, existe unha ampla variedade de proxectos de investigación centrados na utilización de dispositivos BCI para esta finalidade, xa que estes presentan unha gran versatilidade e eficiencia á hora de detectar calquera tipo de actividade cerebral, desde movementos musculares ata emocións, tan só coa análise das ondas cerebrais que capturan [4].

Neste contexto, este traballo pretende desenvolver unha aplicación que, xunto coa utilización dun casco de lectura de electroencefalografías, permita a persoas con discapacidades motrices severas controlar diferentes dispositivos tan só co pestanexo dos seus ollos. En primeiro lugar, o usuario, que levará unha malla de electrodos cunha cámara incorporada, dirixirá a súa mirada cara o dispositivo que queira controlar (movendo a cabeza, se é preciso, para que dito dispositivo quede dentro do campo de visión da cámara), sendo este detectado e recoñecido pola nosa aplicación gracias a librería OpenCV. A continuación, o usuario realizará un número "x" de pestanexos, os cales se verán reflectidos na electroencefalografía que nos proporciona a malla de electrodos previamente mencionada. Esta información será enviada xunto coas imaxes captadas pola cámara, mediante un sistema de comunicación sen fíos, a un dispositivo central que procesará conxuntamente toda a información para obter o número de pestanexos e o dispositivo/elemento que se detecta nas imaxes. Finalmente, esta combinación de datos estará ligada a unha acción sobre o dispositivo que se quería controlar, a cal será transmitida ao mesmo para que este a leve a cabo.

1.2 Motivacións

A principal motivación deste proxecto é dar un primeiro paso para facer moito máis fácil e levadeira a vida das persoas con problemas severos de mobilidade. Na actualidade, a pesar dos grandes avances que se producen día a día no campo da medicina, moitas persoas ven a súa vida truncada debido a diversas enfermidades ou ao verse envoltas en accidentes. Á consecuencia destes feitos, as persoas poden ver a súa capacidade motriz afectada de gravidade, o que conleva un cambio radical nas súas vidas ao ter que depender, na maioría dos casos, de coidadores ou familiares para realizar calquera acción na súa vida cotiá. Esta realidade pode supoñer unha frustación enorme xa que, de repente, unha persoa pasa a depender dos demais para levar a cabo algo tan simple como apagar ou encender unha televisión, as luces, etc.

1.3 Obxectivo xeral

O que busco con este proxecto é reducir, na medida do posible, eses sentimentos de dependencia e de frustración que unha persoa pode sufrir ao estar envolta nalgún dos contextos que se comentaron no apartado anterior. Dotándoas da posiblidade de levar a cabo certas accións con dispositivos da súa contorna, podemos conseguir que vexan que teñen un grao de autonomía maior, o que conleva unha maior facilidade á hora de adaptarse á súa nova vida. Ademais, tamén se intentará crear mecanismos para que poidan comunicarse co resto de persoas sen que estas teñan que estar ao seu lado constantemente, facendo que o traballo dos seus coidadores sexa moito máis doado. Isto último pode ser realmente útil pois, a maiores dos problemas de mobilidade, con algunhas enfermidades tamén se poden desenvolver problemas na fala, polo que con estes mecanismos a comunicación entre ambos sería moito máis fluída.

1.4 Obxectivos técnicos

Alén do obxectivo xeral do proxecto, tamén se tiveron que establecer unha serie de obxectivos técnicos concretos que se intentarán alcanzar ao longo da realización do mesmo. Estes obxectivos técnicos son:

- Comprender os principios básicos de detección e recoñecemento de obxectos mediante a utilización da librería OpenCV e redes neuronais. A librería OpenCV contén unha gran variedade de posibilidades para a súa utilización na detección e recoñecemento de obxectos, polo que constitúe un punto de partida perfecto para a comprensión dos principios básicos deste campo.

- Detectar obxectos domésticos básicos (televisións, mandos a distancia, móbiles, vasos, etc.) a partir de imaxes e fotogramas, facendo uso dos coñecementos adquiridos previamente con OpenCV.
- Estudar as propiedades básicas das ondas cerebrais, e implementar algún tipo de técnica para detectar pestanexos a partir delas.
- Analizar e integrar as diferentes tecnoloxías e protocolos de comunicación sen fíos para poder establecer a conexión entre os diferentes dispositivos/módulos que están involucrados no proxecto.
- Implementar un prototipo software completo e funcional que permita controlar obxectos domésticos mediante pestanexos, validando a súa viabilidade para unha futura implementación en contornas reais.

Compre resaltar este último punto, xa que o obxectivo principal é facer unha primeira aproximación software para validar a viabilidade técnica do produto, permitindo así que de maneira posterior se poida implementar un prototipo hardware.

1.5 Estrutura da memoria

A continuación exponse o contido de cada capítulo do que se compón a memoria:

1. **Introdución:** Neste capítulo introdúcese o proxecto, descríbense os motivos para levalo a cabo e explícanse os obxectivos do mesmo.
2. **Conceptos:** Explícanse todos os conceptos clave cuxa comprensión é imprescindible para a elaboración e comprensión do proxecto.
3. **Fundamentos tecnolóxicos:** Descrición de todas as tecnoloxías e ferramentas utilizadas na elaboración do proxecto.
4. **Metodoloxía, planificación e análise económico:** Exposición da metodoloxía escollida, da planificación inicial e dos recursos utilizados para levar a cabo o proxecto.
5. **Visión xeral do proxecto:** Análise global das diferentes accións que a aplicación ten que levar a cabo para alcanzar o seu obxectivo final.
6. **Detección de obxectos:** Descrición do funcionamento e da elaboración do módulo encargado da detección dos obxectos.
7. **Detección de pestanexos:** Descrición do funcionamento e da elaboración do módulo responsable da detección de pestanexos a partir da Electroencefalografía (EEG) recibida.

8. **Conectividade:** Descrición da implementación e do funcionamento do módulo de conectividade. Expoñerase a tecnoloxía utilizada, como se integrou no entorno de proxecto, etc.

9. **Dispositivo central:** Explicación do funcionamento do módulo de procesamento central, o cal será o principal responsable do funcionamento da aplicación.

10. **Conclusións e traballo futuro:** Exposición das conclusións extraídas tras a elaboración do proxecto e das liñas de traballo para o futuro.

Capítulo 2

Conceptos básicos

Ao longo deste proxecto vanse desenvolver diferentes módulos cuxa implementación está baseada nunha serie de conceptos clave cos que é necesario familiarizarse para entender as decisións de deseño e diferentes aspectos da implementación destes módulos. Polo tanto, neste apartado vanse explicar de forma breve e didáctica estes conceptos clave que aparecerán ao longo do proxecto para facilitar a comprensión do resto da memoria.

2.1 Conceptos relacionados coa detección de obxectos

Para levar a cabo este proxecto necesítase implementar un método efectivo para o recoñecemento e detección de obxectos por parte da aplicación que se vai a desenvolver. Existen varios algoritmos para levar a cabo a detección de obxectos, os cales poden ser divididos en 2 grupos:

- **Algoritmos de clasificación:** Este tipo de algortimos implementan unha estratexia de detección "*two-stage*", e isto provoca que sexan considerablemente lentos, xa que se necesitan realizar predicción para cada rexión seleccionada. Algúns dos exemplos máis coñecidos deste tipo de algoritmos son *Regions with Convolutional Neural Networks* (R-CNN) e todas as súas variantes.
- **Algoritmos de regresión:** Este tipo de algoritmos implementan unha estratexia de detección "*one-stage*", polo que predín clases de obxectos nunha imaxe completa nun só paso e, por conseguinte, son moito máis rápidos que os algoritmos que utilizan a estratexia "*two-stage*". Os algoritmos máis coñecidos deste tipo son *Single Shot Multibox Detector* (SSD) e *You Only Look Once* (YOLO).

Neste proxecto vaise facer uso do algoritmo YOLO [5], un algoritmo de regresión que permite a detección de obxectos en tempo real mediante a utilización dunha única rede neuronal convolucional. Para o seu funcionamento, a rede neuronal divide a imaxe ou frame no que

se queira levar a cabo a detección en rexións, predecindo cadros de identificación e probabilidades por cada rexión, e ponderando se a detección dun obxecto é válida en función da probabilidade asociada.

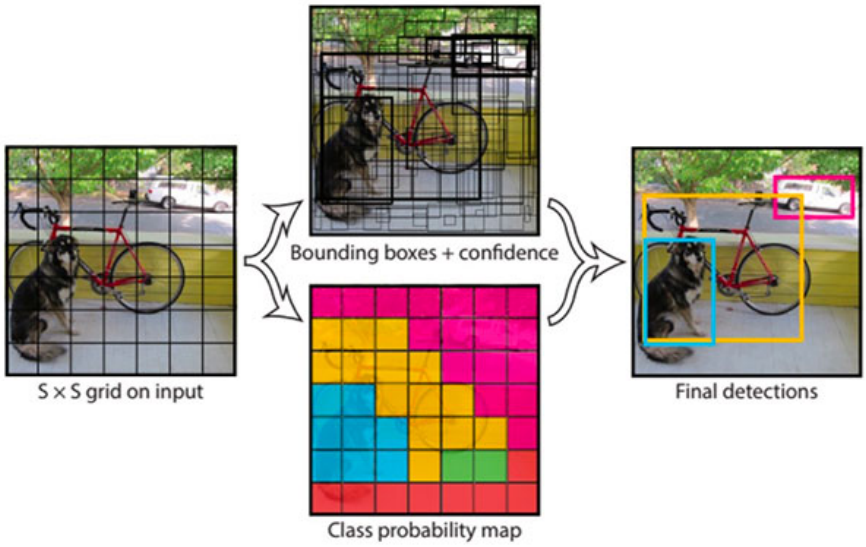

Figura 2.1: Funcionamento do algortimo YOLO [1].

Compre decir que existen múltiples implementacións diferentes deste algoritmo, e que, para que este consiga levar a cabo as deteccións dos obxectos, o algoritmo ten que ser adestrado previamente cun *dataset* apropiado.

A continuación vanse a aclarar algúns conceptos utilizados nesta sección:

- **Redes neuronais:** Son un modelo computacional, inspirado no funcionamento das redes neuronais biolóxicas, que consiste nun conxunto de unidades de procesamento, denominadas neuronas artificiais, que son organizadas en capas e interconectadas entre si. As neuronas da capa de entrada reciben unha serie de datos de entrada, e estes son procesados mediante a realización de operacións lineais facendo uso do peso (un valor númerico) que ten asignado a neurona en cuestión. O resultado obtido en cada neurona é enviado as neuronas da seguinte capa como valor de entrada, seguindo este proceso ata o final da rede. En cada capa da rede, as neuronas integran a información que reciben da/s capa/s anteriores de acordo aos pesos das conexións, e empregan unha función de activación non lineal para introducir non linealidades no procesamento. Este é o modelo máis habitual de redes de neuronas, se ben tamén poden existir outras variantes que implican a retroalimentación de información de capas posteriores e que complican significativamente o deseño. Na figura 2.2 móstrase un esquema básico dunha rede neuronal.

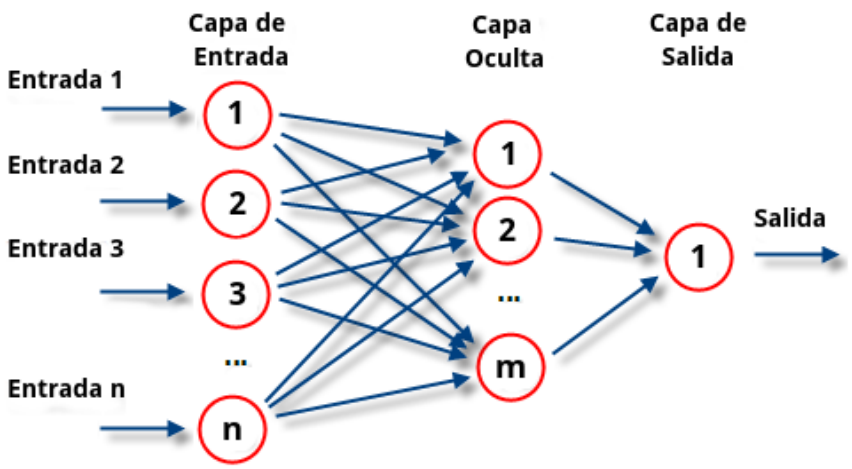

Figura 2.2: Esquema dunha rede neuronal [2].

Para conseguir que unha rede neuronal realice unha funcionalidade concreta, esta necesita ser adestrada de maneira previa. O adestramento dunha rede neuronal consiste na modificación dos pesos das súas neuronas para conseguir un resultado desexado. Para levar isto a cabo, utilízase un método coñecido como *Backpropagation*, o cal consiste en introducir unha serie de datos de adestramento na rede e modificar os pesos en función do erro obtido, e de canto contribuíu cada neurona na obtención do resultado final.

- **Redes neuronais convolucionais:** As redes neuronais convolucionais son un tipo de redes neuronais artificiais donde as neuronas corresponden a campos receptivos dunha maneira moi similar ás neuronas na corteza visual primaria dun cerebro biolóxico [6]. Na práctica, estas redes contan cunha serie de capas ocultas que realizan unha convolución dos datos de entrada cunha serie de *kernels* ou filtros definidos especificamente para extraer certas características dos datos de entrada. Estas capas convolucionais son normalmente seguidas de capas que seleccionan a información relevante. Polo tanto, e debido a que a súa aplicación é realizada mediante matrices bidimensionais, este tipo de redes son moi efectivas para tarefas de visión artificial, como clasificación e segmentación de imaxes entre outras aplicacións.

  Este tipo de redes pódense ver como unha extensión do perceptrón multicapa pero formadas por múltiples capas de filtros convolucionais de unha ou máis dimensións. Ao principio teñen unha fase de extracción de características usando neuronas convolucionais, despois hai unha reducción por muestreo ou reducción de dimensionalidade e, ao final, teranse neuronas de perceptrón máis sinxelas para realizar a clasificación final sobre as características extraídas.

Ao igual que calquera rede neuronal, as redes neuronais convolucionais necesitan ser adestradas de maneira previa ao seu uso. Para iso necesitase facer uso dunha cantidade importante de imaxes de cada obxecto que se queira recoñecer e detectar para que así a rede sexa capaz de extraer as características propias de cada un.

- **Estratexias de detección de obxectos "*one-stage*" e "*two-stage*":** A estratexia de detección "*one-stage*" consiste en intentar detectar directamente sobre a imaxe entrante unha serie de obxectos. Pola contra, os detectores "*two-stage*" realizan dous pasos, no primeiro extraen unha serie de zonas da imaxe na que o detector considera que pode haber obxectos a detectar, e despois realiza un segundo paso no que se intenta detectar os obxectos nas zonas da imaxe que se obtiveron previamente. Comunmente, os "*two-stage*" obteñen resultados máis fiables, pero son moito máis lentos que os "*one-stage*".

- **Dataset:** Colección de datos, que no caso deste proxecto será unha colección de imaxes, que se utilizará no módulo de detección de obxectos para adestrar ao detector. Os datos que forman parte desta colección deben incluír o conxunto de datos de entrada que se lle pasará á rede neuronal (imaxes) e o resultado que esta debe obter (lista de obxectos detectados). É fundamental que o *dataset* conteña exemplos representativos de todos os posibles datos de entrada que nos interese considerar, coas súas correspondentes solucións, para o problema que se pretende resolver.

- **Modelo preadestrado:** Para que o detector de obxectos que se vai a utilizar neste proxecto consiga detectar unha serie de obxectos determinados, este ten que ser adestrado utilizando un *dataset* axeitado. Estes procesos de adestramento poden conlevar unha gran carga computacional polo que, neste proxecto, decidiuse utilizar un modelo preadestrado, é decir, un detector que xa foi adestrado e está listo para a súa utilización de cara a detectar obxectos presentes no *dataset* co que foi adestrado.

- **PSNR (do inglés, Peak Signal-to-Noise Ratio):** Termo utilizado en enxeñaría para definir a relación existente entre a máxima enerxía dun sinal e o ruido que afecta a súa representación fidedigna [7]. O seu uso máis habitual é como medida cuantitativa da calidade de reconstrucción no ámbito da compresión de imaxes, pero no caso deste proxecto vaise a utilizar como medida cuantitativa da varianza existente entre dúas imaxes, o que nos permitirá comprobar se existe moito movemento entre dous frames dun vídeo a procesar. Matematicamente, esta métrica calcúlase do seguinte xeito:

$$\text{psnr} = 10\log_{10}\left(\frac{p_{\text{max}}^2}{\text{mse}}\right) = 20\log_{10}(p_{\text{max}}) - 10\log_{10}(\text{mse}), \tag{2.1}$$

onde p_{max} representa o valor máximo que pode tomar un píxel da imaxe e ven dado por $2^b - 1$ con b sendo a profundidade de bit, e o termo mse representa o error cuadrático

medio entre as dúas imaxes ou *frames* que se están a comparar, é dicir:

$$\text{mse} = \frac{1}{MN} \sum_{i,j} \Big[A(i,j) - B(i,j) \Big]^2, \tag{2.2}$$

considerando que as dúas imaxes que se están a comparar A e B teñen unha dimensión de $M \times N$ pixels.

- TP é o número de verdadeiros positivos, que neste caso simbolizaría o número de obxectos que aparecen nos vídeos e que foron correctamente detectados.
- FN é o número de falsos negativos, que neste caso simbolizaría o número de obxectos presentes nos vídeos que non foron detectados polo detector.

2.2 Conceptos do detector de pestanexos

Para comprender o funcionamento do detector de pestanexos, compre entender previamente unha serie de conceptos sobre os datos cos que imos traballar, polo que nesta sección vanse a explicar unha serie de conceptos sobre os sinais dos EEG:

2.2.1 Que é un EEG

EEG son unhas siglas que fan referencia ao termo Electroencefalografía, o cal consiste na representación gráfica dun proceso electrofisiolóxico no cal se recopila a actividade eléctrica do cerebro.

2.2.2 Como se obteñen

Os sinais das EEG obtéñense mediante a realización de test/análises de electroencefalografías, os cales consisten en colocar unha serie de sensores, uns pequenos discos metálicos que tamén se denominan electrodos, na cabeza dun suxeito (figura 2.3). Os sinais EEG que se conseguen a través destes sensores son amplificados, dixitalizados e enviados a un dispositivo, no cal se almacenarán ou serán precesados de distintas formas en función dos datos que se queiran obter a partir deles.

2.2.3 Sinais/ritmos presentes nas EEG

As células que conforman o cerebro humano xeran constantemente unha serie de sinais eléctricos que forman patróns non linais, e estes son definidos como ondas ou ritmos cerebrais. Estas ondas representan unha gran variedade de información e son clasificadas en función da súa frecuencia en 5 grandes tipos (figura 2.4) [8] :

Figura 2.3: Dispositivo utilizado para a abtención de sinais EEG.

- **Ondas Gamma:** O seu rango de frecuencias vai dos 30 aos 100 Hz. En persoas adultas e sás, a presenza de ondas gamma na actividade cerebral está relacionada con algunhas funcións motrices (sobre todo durante a contracción de músculos) ou percepcións.
- **Ondas Betha:** O seu rango de frecuencias vai dos 14 aos 30 Hz. Entán comunmente asociadas a estados de conciencia, alerta ou atención. Tamén o están con estados mentais, como a concentración ou a ansiedade, e con decisións motrices, como a supresión do movemento ou a sensación de movemento.
- **Ondas Alpha:** O seu rango de frecuencias vai dos 7 aos 13 Hz, ainda que en adultos pode variar a rangos de 9 a 11 Hz. A miúdo asócianse a estados mentais de relaxación, calma e lucidez, polo que poden inducirse simplemente con pechar os ollos e relaxarse, e non estarán presentes durante procesos cognitivos intensos (por exemplo, durante cálculos mentais).
- **Ondas Theta:** O seu rango de frecuencias vai dos 4 aos 7 Hz. En persoas novas son relacionadas con frecuencia a que o individuo está hiperventilando. Pola contra, en persoas máis maiores, son moito máis atípicas de ver, agás en estados de somnolencia.
- **Ondas Delta:** Están caracterizadas por frecuencias baixas (3 Hz) e amplitudes de onda grandes. Poden ser asociadas con etapas de insomnio e non son moi usuais en adultos que estean espertos e non presenten ningún problema relacionado con algún tipo de desorde neurolóxico.

2.2.4 Artefactos

As EEG foron incialmente deseñadas para o rexistro da actividade cerebral dun individuo, pero os electrodos que se utilizan para realizar este cometido son capaces de detectar outro tipo de actividades eléctricas de orixe non cerebral, as cales son denominadas artefactos [9].

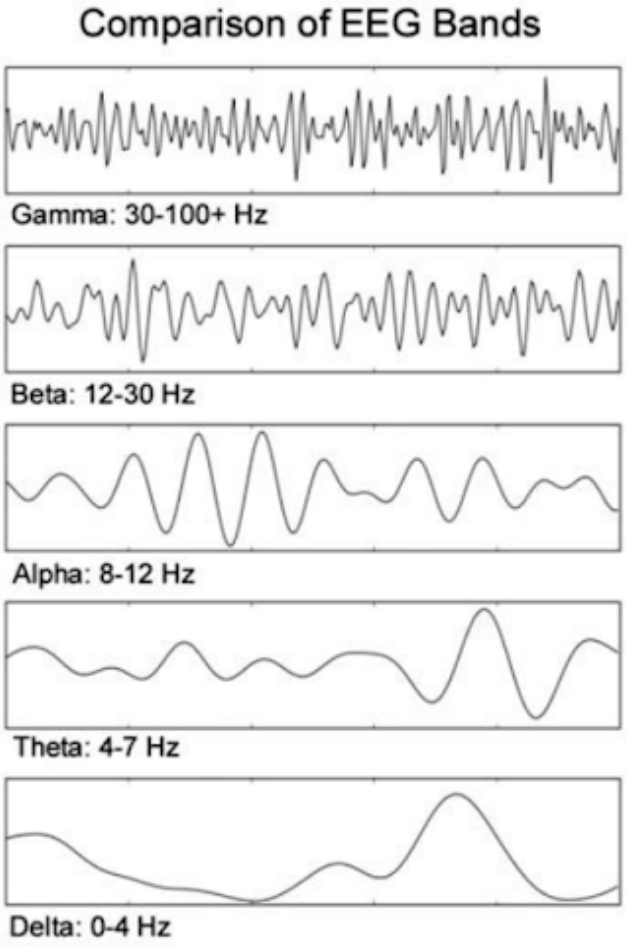

Figura 2.4: Ritmos ou ondas cerebrais.

Estes artefactos pódense dividir en dous tipos en función do seu orixe: artefactos relacionados cos suxeitos (asociados a movementos, alteracións de distinta índole, etc) e artefactos técnicos (asociados a movementos de cables, alimentación, etc), polo que teñen que ser xestionados de maneira diferente. A continuación mostraranse algúns dos artefactos máis comúns [10]:

- **Artefactos provocados por movementos oculares:** Os globos oculares actúan como un dipolo, coun polo positivo (córnea) e un polo negativo (retina). Cando o globo rota sobre o seu eixe xera un campo de corrente alterna de gran amplitude que é detectable por parte dos electrodos situados preto do ollo. Un pestenxo causa que o polo positivo (a córnea) se achegue aos electrodos frontopolares, producindo deflexións simétricas hacia abaixo. Esto dá lugar a aparición dun patrón recoñecible nos sinais de EEG.
- **Artefactos da pel:** Aparece unha dificultade adicional debido ás propiedades de certas capas da pel. Existe un potencial de corrente continua significativo entre o estrato córneo e o estrato granuloso, e calquera deformación local da pel alterará este potencial. A única forma confiable de eliminar a fonte do artefacto é crear unha vía de baixa resistencia a través das capas da pel mediante a limpeza da mesma cun hisopo con alcohol.
- **Artefacto dos 50Hz:** O problema xorde cando a impedacia dun dos electrodos activos pasa a ser significativamente grande entre os electrodos e a terra do amplificador. Nesta situación, a terra pasa a ser un electrodo que, dependendo da súa localización, produce un artefacto na banda dos 50Hz. A interferencia da radiación de alta frecuencia de outros dispostivos electrónicos pode ademais sobrecargar os amplificadores da EEG, extendendo este efecto.

Cando se intenta analizar un EEG, un dos pasos máis importantes é a detección e eliminación dos artefactos, xa que os sinais cerebrais son moi débiles e vense moi influenciados polos artefactos e ruídos que poidan interferir na gravación. Sen embargo, no caso deste proxecto, vaise a facer todo o contrario. O que se busca neste proxecto é localizar os artefactos provocados polos pestanexos para contabilizar cantos se realizan nun período de tempo, polo que este tipo de artefactos relacionados co suxeito pasan a ser a información de valor que se busca nos EEG. Compre resaltar que estes artefactos provocados por pestanexos aparecerán en frecuencias baixas, polo que alterarán as actividades cerebrais producidas nos ritmos Tetha e Delta xa que, tal e como se comentou previamente, son os ritmos que aparecen nos rangos de frecuencias máis baixos.

2.2.5 Que mostran os EEG

Tal e como se pode apreciar na figura 2.5, as EEG mostran a actividade cerebral dun individuo [11]. Esta activade cerebral representa as ondas eléctricas que se producen no interior do seu

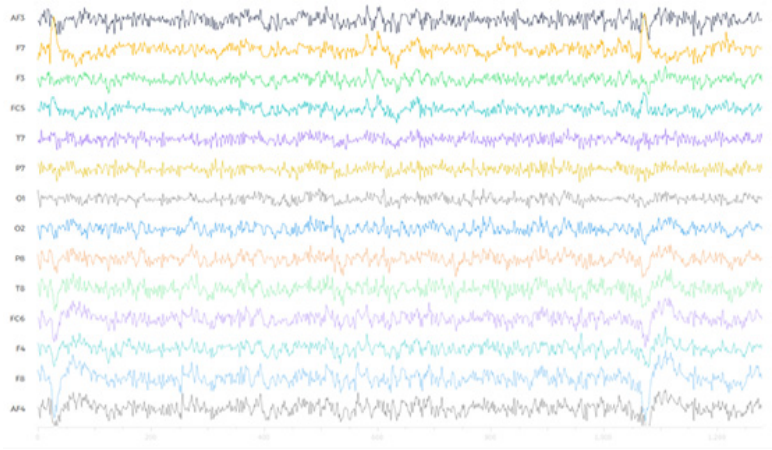

Figura 2.5: Exemplo de EEG.

cerebro, as cales son dixitalizadas e mostradas a través dun dispositvo de EEG e un BCI. Nestes sinais, a amplitude indica o valor en microvoltios (μV) das diferentes correntes eléctricas que se producen para cada instante de tempo.

Nun principio, os sinais eléctricos recollidos polos electrodos son continuos no dominio do tempo. No proceso de dixitalización, realízase unha mostraxe que consiste en coller o valor que toma o sinal en intervalos regulares de tempo. Estos intervalos dependen da frecuencia de muestreo do dispositivo BCI e a precisión dos valores da amplitude dependen da resolución do mesmo. Polo tanto, despois desta dixitalización, o sinal de EEG representará un conxunto de mostras discretas en tempo coa súa correspondente amplitude medida en μV.

Agora que xa se sabe que representan as EEG, imos centrarnos na información que se quere extraer das mesmas. No caso deste proxecto, o que se busca é detectar os pestanexos voluntarios dunha persoa a partir dos seus sinais de EEG. Como xa se anticipou, os artefactos causados polos pestanexos nas EEG seguen un patrón bastante recoñecible que se pode observar na figura 2.6. Estes patróns son un artefacto propio das EEG e, tal e como se pode apreciar, presentan uns picos de amplitude moito máis grande que o resto do sinal.

2.2.6 Filtros

Un filtro eléctrico ou electrónico pode definirse como un elemento que discrimina unha determinada frecuencia ou gama de frecuencias dun sinal eléctrico que pasa a través del, podendo modificar tanto a amplitude como a súa fase [12]. Estes filtros poden ser clasificados en:

- Activos ou pasivos.

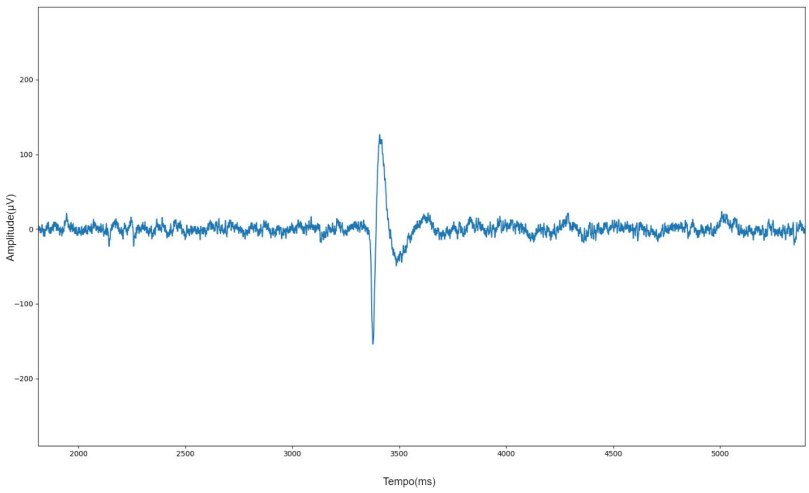

Figura 2.6: Patrón dun pestanexo.

- Analóxicos ou dixitais.
- De paso alto, paso baixo, paso banda,...
- De tempos discretos ou continuos.
- De resposta infinita ou finita ao impulso.

Neste proxecto en concreto vaise facer uso dun filtro paso banda [13], o cal consiste nun tipo de filtro electrónico que deixa pasar un determinado rango de frecuencias e atenúa o paso do resto. O obxectivo pricipal polo cal se vai a utilizar este filtro é eliminar tanto o ruído concentrado en certas bandas de frecuencia do sinal que non son de interese como a compoñente de continua do mesmo. Na figura 2.7, pode verse a resposta en frecuencia que presenta un filtro paso banda. Como pode apreciarse, un filtro destas características atenúa as compoñentes de frecuencia do sinal de entrada que están fóra da banda limitada polas frecuencia de corte f_1 e f_2.

2.2.7 Compoñente de continua dun sinal

A compoñente de continua dun sinal correspóndese co valor medio do mesmo e coas compoñentes de frecuencia do sinal en 0 Hz. Desta forma, esta compoñente non aporta información relevante cando se analizan patróns de variación sobre un sinal, xa que simplemente se pode ver como un valor constante que se suma a todas as mostras do sinal. Por esa razón, resulta máis cómodo eliminar esta compoñente e facer a análise sobre o sinal resultante.

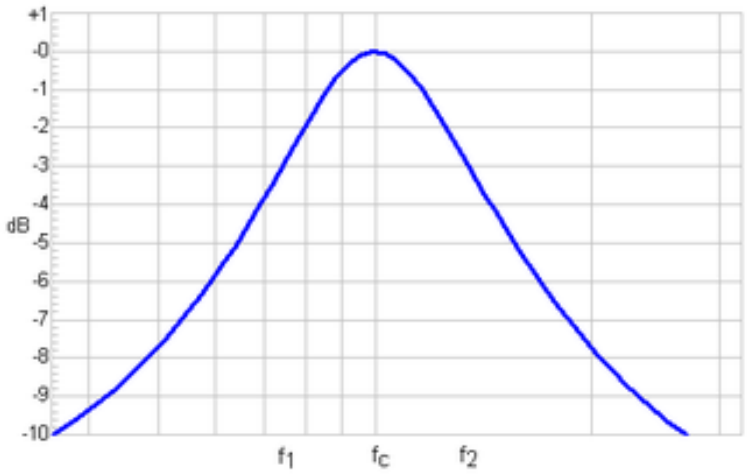

Figura 2.7: Resposta frecuencial dun filtro paso banda.

2.3 Conceptos de conectividade

Para a realización deste proxecto é necesario utilizar unha tecnoloxía ou protocolo de comunicación para interconectar os diferentes dispositivos que se vaian a utilizar. Existe unha gran variedade de tecnoloxías e protocolos pero, para este proxecto, optouse por utilizar o protocolo de comunicación *MQ Telemetry Transport* (MQTT) [14], o cal está gañando unha gran popularidade debido ao seu uso no campo do *Internet of Things* (IoT).

- **IoT:** Sistema de interconexión de dispositivos dixitais de uso cotiá que proporciona a capacidade de transferir información a través da rede formada polos dispositivos sen requerir ningún tipo de interacción humano a humano ou humano a ordenador [15]. A comunicación existente entre os diferentes dispositivos que pertencen a rede de IoT pode variar dependendo do protocolo da comunicación que se utilice (AMQP, WAMP, MQTT, ...), xa que non hai un estándar único definido para este tipo de sistemas. Ainda así, todos estes protocolos presentan unha serie de características comúns, entre as que podemos encontrar as seguintes:

  - Seguen unha estratexia de comunicación M2M (*Machine to Machine*).
  - As mensaxes adoitan ter un tamaño bastante limitado.
  - Son protocolos lixeiros centrados na comunicación sen fíos.

  Compre resaltar que, debido a que utilizan mensaxes cun tamaño basatante limitado, a maioría deste protocolos de comunicación presentan problemas á hora de intercambiar arquivos dun tamaño considerable. Por exemplo, MQTT limita o tamaño do contido das

súas mensaxes (denominado *payload*) a 256 MB, polo que no caso de que se queira enviar un arquivo superior a este tamaño límite, a única posibilidade sería trocear o arquivo en trozos de 256 MB e enviar ditos trozos a través de varias mensaxes consecutivas.

A aplicación desenvolvida neste proxecto segue unha arquitectura e a filosofía clásica dos sistemas de IoT, coas súas correspondentes limitacións. Desta forma, os protocolos de comunicación deseñados para estos sistemas resultan ventaxosos e axeitados para a implementación do módulo de comunicacións requerido.

- **Patrón publicador/subscriptor:** Patrón de mensaxería donde un axente, denominado subscritor, informa a un servidor central de que quere recibir todas as mensaxes que cheguen ao servidor cunha temática determinada, cuxo término específico sería *topic*, e outro axente, denominado publicador, publica nun servidor mensaxes cunha temática asociada (figura 2.8) [16].

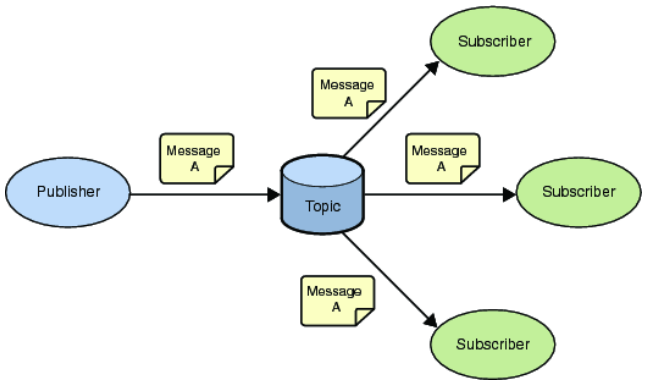

Figura 2.8: Patrón de mensaxería publicador-subscritor.

- **QoS (Quality of Service):** Mecanismo utilizado para controlar o tráfico dunha rede que permite, entre outras cousas, priorizar paquetes para mellorar a fluidez da transmisión na rede ou garantizar a entrega dunha mensaxe específica [14].

Capítulo 3

Fundamentos tecnolóxicos

Para a realización do proxecto, tanto a nivel de implementación como a nivel de planificación, vanse necesitar unha serie de tecnoloxías e ferramentas software. Para comprender o uso que se fará delas, é necesario entender previamente en que consisten e en que fundamentos tecnolóxicos se basan. Polo que ao longo deste capítulo explicaránse os fundamentos e a funcionalidade destas ferramentas e tecnoloxías.

3.1 Python

Python é unha linguaxe de programación de alto nivel, interpretado e de propósito xeral que foi creada por Guido van Rossum [17]. A filosofía principal na que se basea esta linguaxe é facer fincapé na claridade do código, facendo que os programadores poidan crear códigos lexibles e lóxicos para proxectos de calquera tamaño. Entre as suas principais características Compre resaltar as seguintes:

- É unha linguaxe de programación multiparadigma, xa que soporta varios paradigmas de programación, como a programación orientada a obxectos, a programación funcional ou a estruturada.
- Python utiliza tipado dinámico para a administración da memoria, o que permite que unha variable poida tomar valores de diferentes tipos.
- Pode ser executado en varios sistemas opertivos diferentes e de forma libre, xa que se trata de código aberto.

Neste proxecto optouse pola utilización de Python coma linguaxe de programación por diversos motivos. En primeiro lugar, pola súa claridade e simplicidade á hora de programar, o que permite unha aprendizaxe e adaptación moi rápida en comparación con outras linguaxes de programación como C ou Matlab. Outro dos puntos fortes de Python é a súa portabilidade, xa

que pode ser executado en diversos sistemas de maneira sinxela e sen necesidade de realizar instalacións longas ou farragosas, o que resulta moi útil neste proxecto debido a que se traballará con diversos sistemas como as BCIs, ordenadores, etc. Finalmente, compre destacar que Python proporciona moitas utilidades que simplifican considerablemente o tratamento e procesado de sinais, as operacións matemáticas sobre vectores, a implementación de mecanismos de comunicación e é compatible con OpenCV, polo que é unha linguaxe perfecta para a realización deste proxecto.

3.2 OpenCV

OpenCV é unha librería de visión artificial que fai uso do *machine learning* para permitirnos detectar e recoñecer obxectos entre outras moitas cousas [18]. Esta librería está implementada en C, pero soporta o seu uso con outras linguaxes de programación, polo que neste proxecto vaise a utilizar Python para desenvolver a aplicación. Á hora de utilizar OpenCV para o recoñecemento de obxectos hai que prestar atención aos diversos mecanismos que a librería nos proporciona, xa que existen diferentes métodos e resulta importante seleccionar o máis axeitado.

Dentro de OpenCV, podemos destacar 3 tipos de algoritmos de detección de obxectos:

- R-CNN (*Regions with Convolutional Neural Networks*) e todas a súas variantes.
- SSD (*Single Shot Multibox Detector*).
- YOLO (*You Only Look Once*).

Os detectores creados con OpenCV presentarán un rendemento moi diferente en función das características do algortimo de detección utilizado, polo que é moi importante prestar atención a cal se ten que utilizar en función das prioridades de cada un. Unha das características máis importantes é a estratexia de detección propia de cada algortimo. O algoritmo R-CNN segue unha estratexia "*two-stage*" e os algortimos SSD e YOLO seguen unha estratexia "*one-stage*", polo que R-CNN obtén resultados máis fiables con respecto aos outros algoritmos, pero esta ventaxa obtena a cambio de conlevar unha carga computacional máis grande e ser moito máis lento ca SSD e YOLO. Tendo isto en conta, neste proxecto optouse por utilizar o algortimo de detección YOLO debido a que, aínda que é menos fiable ca R-CNN, prímase a velocidade do detector por encima do nivel de fiabilidade, xa que se a aplicación se executa en tempo real que o detector tardase moito tempo en executarse sería lago inaceptable.

Por outra parte, os detectores serán os encargados de recoñecer os obxectos do entorno do usuario da nosa aplicación, pero para que estes consigan detectar unha serie de dispositivos/obxectos teñen que ser adestrados cun dataset apropiado. O dataset que se escolla para adestrar o detector ten que estar formado por unha boa cantidade de fotos dos obxectos que se buscan detectar para que o detector sexa adestrado correctamente, e así minimizar a taxa de fallos nas deteccións. No noso caso vaise facer uso do *dataset* COCO [19] para adestrar o detector YOLOv3 que se vai a utilizar, xa que este dataset contén unha gran variedade obxectos que se queren detectar mediante a nosa aplicación (televisións, móbiles, mandos a distancia, ...).

3.3 BCI

Para levar a cabo a lectura dos impulsos eléctricos que se producen no noso cerebro e obter os sinais de EEG correspondentes, a partir dos cales se pode establecer se a persoa realizou unha serie de pestanexos, imos facer uso dunha interfaz cerebro-ordenador (BCI, do inglés *Brain Computer Interface*). Estes dispositivos permiten crear unha vía de comunicación bidireccional e directa entre o cerebro do usuario e un ordenador, enviando a través desta vía información acerca da actividade cerebral do usuario ao ordenador para que este a decodifique, ordene e utilice en función do uso que se lle queira dar.

Para a realización deste proxecto fíxose uso do hardware OpenBCI Cyton Biosensing Board (figura 3.1) [20], unha interface cerebro-ordenador que conta con 8 canais para a conexión de electrodos e cun procesador de 32 bits. Ademáis, leva incorporado un microcontrolador PIC32MX250F128B que lle proporciona unha gran cantidade de memoria local e unha velocidade de procesamento moi alta, o *firmware* máis actual de OpenBCI e o *bootloader* chipKIT. Este dispositivo pode usarse para a detección de actividade cerebral (EEG), muscular (EMG) e cardíaca (ECG), polo que presenta unha gran versatilidade á hora de detectar diferentes tipos de sinais. Compre resaltar que toda a información procesada pola interface será mostrada a unha frecuencia de 250Hz en cada un dos 8 canais.

Unha vez están aclaradas as características hardware da interface que se vai a utilizar, é necesario aclarar como se conectan os diferentes electrodos á placa e como se van a utilizar para a realización das gravacións. Os distintos pares de electrodos son combinados formando montaxes. Hai dous tipos básicos de montaxes[21]: bopilar (transversal e lonxitudinal) e monopolar (o referencial). O bipolar rexistra a diferencia de voltaxe existente entre os electrodos colocados en áreas de actividade cerebral, mentras que o monopolar rexistra a diferencia de potencial entre un electrodo ubicado nunha zona cerebral activa e outro colocado sobre unha área sen actividade ou neutra; ou ben a diferencia de voltaxe entre un electrodo colocado

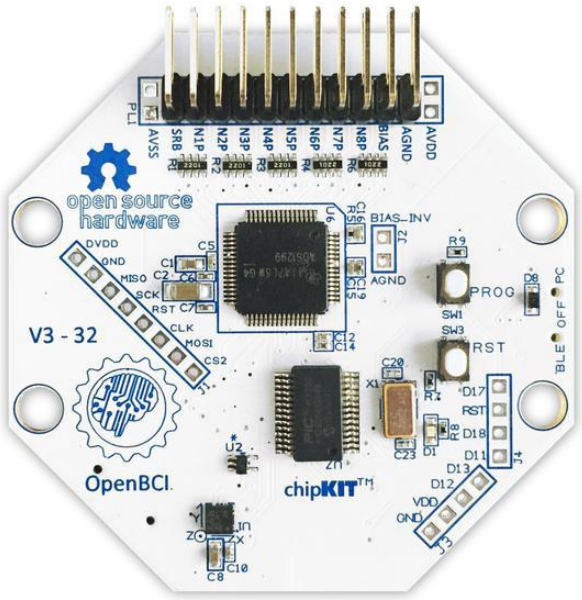

Figura 3.1: Hardware OpenBCI Cyton Biosensing Board.

nunha zona activa e o promedio de todos ou algúns dos electrodos activos. O noso BCI fai uso dunha montaxe monopolar, na cal se van a utilizar un electrodo de entrada (o cal se conectará a través dun dos 8 canais), un electrodo de referencia e un de terra (estes dous últimos están integrados na propia placa do BCI). A continuación vaise mostrar unha imaxe (figura 3.2) de como se poderían distribuir os electrodos no coiro cabeludo e explicarase para que serve e en que lugar se colocará cada electrodo, xa que no caso deste proxecto non se seguirá a distribución que se mostra na imaxe. [22].
No noso caso a distribución que se vai a seguir será a seguinte:

- **Electrodo de entrada:** Na imaxe referénciase como VIN+. Este electrodo é o encargado de detectar a actividade cerebral que se quere recoller e será colocado na zona pre-frontal sobre o ollo dereito.

- **Electrodo de referencia:** Na imaxe referénciase como VIN-. Este electrodo, que será colocado no mastoide dereito (detrás da orella dereita), encárgase de recoller o voltaxe existente na zona na que é colocado, o cal será utilizado xunto coa información recollida polo electrodo de entrada para a obtención da EEG.

- **Electrodo de terra:** Na imaxe referénciase como GND. Este electrodo, que será colocado no lóbulo da orella dereita, é utilizado para evitar calquera tipo de ruido provocado pola alimentación, o cal interferería na gravación dos sinais de interés.

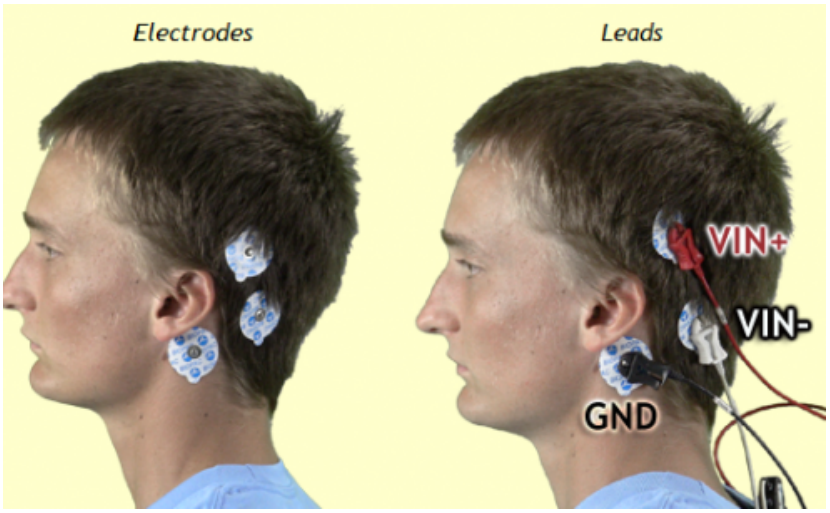

Figura 3.2: Exemplo de distribución dos electrodos.

Finalmente hai que falar da ferramenta software que trae instalado o BCI que se vai a utilizar, a OpenBCI GUI [23]. Esta ferramenta trae implementada a visualización e gravación en arquivos .txt da información que chega a placa do BCI. Tamén trae certas utilidades que permiten ao usuario, entre outras cousas, abrir portos para realizar un *streaming* de datos entre a ferramenta e outro software diferente, procesar a información recibida mediante a aplicación de diferentes tipos de filtros e transformadas para, por exemplo, poder analizar os sinais no dominio da frecuencia, etc.

Figura 3.3: Imaxe de OpenBCI GUI.

3.4 MQTT

Neste proxecto vanse ver implicados varios dispositivos diferentes, polo que é necesario encontrar unha tecnoloxía compatible con ditos dispositivos para poder interconectalos entre si. A día de hoxe, algunhas das tecnoloxías de comunicación máis utilizadas por todo o mundo son Bluetooth e NFC, pero neste proxecto vaise facer uso do protocolo de comunicación MQTT. O motivo principal polo que se escolleu MQTT é que a súa implementación no proxecto é moi sinxela, xa que como a súa implementación se vai a levar a cabo mediante a linguaxe de programación Python pódese facer uso de librerías como paho-mqtt, a cal simplifica de maneira considerable a creación de programas para a utilización deste protocolo. Ademáis, MQTT está baseado na pila TCP/IP, polo que para a súa utilización tan só se necesitan un chip Wi-Fi e os pertinentes paquetes de software nos dispositivos que vaian a utilizar o protocolo en cuestión.

3.4.1 Que é MQTT

MQTT son as siglas de *MQ Telemetry Transport*, e trátase dun protocolo de comunicación M2M (*machine-to-machine*), lixeiro e que segue un patrón publicador/subscritor. Este protocolo foi deseñado por Andy Stanford-Clark e Arlen Nipper en 1999 para conectar sistemas de telemetría de oleoductos por satélite [14] [3] e, ainda que comezou como un protocolo propietario, este foi liberado en 2010, converténdose nun estándar OASIS en 2014 [24].

O protocolo está baseado na pila TCP/IP como base para a comunicación e cada conexión que se establece manténse aberta para a súa reutilización en cada comunicación. Esta característica é unha das suas principais diferencias con respecto a outros protocolos como HTTP 1.0, onde cada transmisión é realizada a través dunha conexión nova.

Compre mencionar que este protocolo estase a converter nun dos protocolos máis utilizados no contexto do IoT, polo que está moi presente en diferentes dispositivos de domótica.

3.4.2 Como funciona MQTT

Unha sesión MQTT pode dividirse en catro etapas: conexión, autenticación, comunicación e desconexión. O primerio paso lévase a cabo cando un cliente crea unha conexión de TCP/IP co *broker/server* utilizando un dos portos establecidos polo *broker*. Os portos estándar son o 1883 para a comunicación non cifrada e o 8883 para a cifrada (con SSL/TLS).

O segundo paso sería a autenticación. Tras establecer a conexión, o cliente terá que autenticarse mediante un certificado ou cun usuario e contrasinal. Sen embargo, este paso non é

estrictamente necesario en todas as situacións. Algúns *brokers* MQTT soportan clientes anónimos, especialmente aqueles *brokers* online que están abertos e dispoñibles para o seu uso de maneira gratuita, polo que non sería necesario ningún proceso de autenticación para establecer a conexión con este tipo de *brokers* MQTT.

O seguinte paso consistiría na comunicación entre os subscritores/publicadores e o *broker* MQTT. Para entender correctamente este apartado, compre coñecer antes a definición e tipoloxía das mesaxes que utiliza o protocolo:

Estructura dunha mensaxe MQTT:

Cada mensaxe consta de 3 partes [3] tal e como se pode apreciar na figura 3.4:

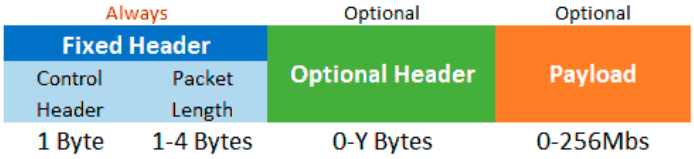

Figura 3.4: Estructura dunha mensaxe MQTT.

- **Cabeceira fixa:** Ocupa de 2 a 5 bytes e trátase dun campo obrigatorio. Consta dun código de control que identifica o tipo de mensaxe eviada (1 byte) e dun campo coa lonxitude da mensaxe, a cal se codifica en 1-4 bytes, dos cales o último bit é un bit de continuidade. Na figura 3.5, móstranse os códigos de control existentes.
- **Cabeceira variable:** Campo opcional que contén información adicional necesaria en algunhas situacións ou mensaxes.
- **Contido (payload):** É o contido real da mensaxe. Pode ter un tamaño máximo de 256 MB polo que, en caso de querer enviar un arquivo maior a 256 MB, sería necesario trocear a información correspondente en e enviar varias mensaxes.

Para establecer a comunicación co *broker* MQTT, o cliente actuará dunha maneira determinada en función do rol que queira desempeñar na rede. En caso de que o cliente queira ser un subscritor, o primeiro paso será enviarlle ao *broker* unha mensaxe de tipo SUBSCRIBE, na cal o cliente indicará o *topic* (nome técnico que fai refernecia a unha temática) ao que se quere suscribir para, dese xeito, recibir todas as mensaxes publicadas con ese *topic* concreto. Compre destacar que estes *topics* representan unha estrutura xerárquica, é decir, un cliente subscrito ao *topic* "casa/salon" recibirá as mensaxes publicadas cos *topics* "casa/salon/television" e "casa/salon/calefacción". Finalmente, o *broker* MQTT enviaralle ao subscritor unha

Message	Code
CONNECT	0x10
CONNACK	0x20
PUBLISH	0x30
PUBACK	0x40
PUBREC	0x50
PUBREL	0x60
PUBCOMP	0x70
SUSBSCRBE	0x80
SUBACK	0x90
UNSUSCRIBE	0xA0
UNSUBACK	0xB0
PINGREQ	0xC=
PINGRESP	0xD0
DISCONNECT	0xE0

Figura 3.5: Códigos de control das mensaxes MQTT [3].

mensaxe SUBACK ou UNSUBACK en función de se a subscripción se realizou con ou sen éxito. Pola contra, no caso de que o cliente queira ser un publicador, este simplemente terá que enviarlle a mensaxe co *topic* coa que quere publicar dita mensaxe.

Outra característica propia da fase de comunicación en MQTT é o envío de mensaxes PINGREQ. Estas mensaxes son enviadas periódicamente polos clientes da rede ao *broker* para asegurarse de que a conexión TCP está activa. Como resposta a estas mensaxes, o *broker* MQTT devolverá unha mensaxe PINGRESP.

A última fase é a de desconexión, na cal se poden dar dúas situacións:

- **Desconexión normal:** Os clientes que queiran desconectarse da rede envían ao *broker* unha mensaxe DISCONNECT e despois pechan sesión. Este tipo de desconexión proporciona ao cliente a capacidade de reconectarse facilmente porporcionando a súa identidade e reanudando a conexión donde a deixou.
- **Desconexión abrupta:** A desconexión prodúcese de maneira abrupta e sen tempo para que o cliente envíe a mensaxe de desconexión.

3.4.3 Calidade do servicio (QoS)

MQTT conta cun mecanismo para establecer requisitos de calidade de servicio ou QoS na transmisión de mensaxes. Este mecanismo é empregado para xestionar a robustez do envío de mensaxes ao cliente fronte a fallos. Polo tanto, MQTT otorga ao cliente a posibilidade de elixir entre 3 niveis de servicio:

- **QoS 0:** É o nivel máis simple e utiliza unha secuencia de mensaxes PUBLISH. Neste nivel de QoS, o publicador envía a mensaxe ao *broker* só unha vez e o *broker* envíaa aos subscritores só unha vez. Non existe ningún tipo de mecanismo de recuperación fronte a fallos, polo que a mensaxe podería non chegar ao seu destino.
- **QoS 1:** Segundo nivel de QoS no que se utiliza unha secuencia de mensaxes PUBLISH/-PUBACK entre o publicador e o *broker* e entre o *broker* e os subscritores. Neste caso, a mensaxe é enviada ao seu destinatario e, de maneira posterior, recíbese unha mensaxe PUBACK que garantiza a súa recepción. Con este sistema garantízase que a mensaxe chega como mínimo unha vez pero, se se produce un fallo no envío/recepción da mensaxe PUBACK, pódese dar o caso de que se reciban mensaxes duplicadas. Este duplicado de mensaxes prodúcese debido a que se, por exemplo, o publicador non recibe un PUBACK tras a publicación dunha mensaxe este interpreta que o broker non recibíu,

polo que volve a enviala ata recibir a mensaxe de confirmación de recepción PUBACK por parte do broker.

- **QoS 2:** Terceiro nivel de QoS que tamén se denomina nivel de servicio asegurado. Neste nivel entréganse dous pares de mensaxes, o primeiro denomínase PUBLISH/PUBREC e o segundo PUBREL/PUBCOMP. Estes dous pares de mensaxes aseguran que, independentemente do número de reintentos, a mensaxe entregarase só unha vez, evitando así o problema que presentaba o nivel anterior cando se recibía mensaxes duplicadas en caso de fallo.

3.5 Git

Un punto clave para a elaboración dun proxecto deste tipo é a elección dunha ferramenta de control de versións. Para realizar este proxecto escolleuse a ferramenta Git[25], a cal nos permite, entre outras cousas, levar un histórico claro das diferentes etapas do proxecto ou crear dunha maneira sinxela diferentes ramas do mesmo, o cal resulta moi útil á hora de realizar probas sobre diferentes versións. En concreto, fíxose uso desta ferramenta mediante a utilización do repositirio virtual de GitLab que a FIC proporciona a cada alumno [26].

Compre resaltar que Git tamén resulta tremendamente útil debido a que permite ter dispoñible en todo momento os arquivos do proxecto nun repositorio remoto, podendo acceder a estes facilmente para traballar en local. Esta funcionalidade é especialmente útil cando o desenrolo do proxecto é realizado por un equipo de traballo, xa que todos os membros do equipo teñen dispoñibles os cambios e actualizacións levadas a cabo polo resto de compañeiros unha vez estes os suban ao repositorio remoto do proxecto.

3.6 Latex

Para levar a cabo a redacción desta memoria fíxose uso de LaTeX [27], un sistema de composición de textos que está formado maioritariamente por ordes construidas a partir de comandos de TeX e que é usado comúnmente para a elaboración de artículos e libros de carácter científico. En concreto, fíxose uso do servizo online de Overleaf [28], o cal permite a creación e elaboración de documentos mediante o uso de LaTeX e do modelo de memoria que é proporcionado por parte da FIC para a elaboración dos TFG.

3.7 Project Libre

Project Libre é unha ferramenta para a administración e xestión de proxectos de código aberto, moi similar ao software de pago Microsoft Project. Esta ferramenta foi lanzada en 2012 para ofrecer unha alternativa gratuíta a Microsoft Project. Neste proxecto, este programa foi utilizado para crear unha planificación axeitada para o desenvolvemento do proxecto e para levar a cabo unha análise económica do mesmo.

Capítulo 4

Metodoloxía, planificación e análise económico

4.1 Metodoloxía

Para o desenvolvemento deste proxecto escolleuse, entre a gran variedade de metodoloxías existentes, unha metodoloxía iterativa incremental [29], a cal consiste en agrupar as tarefas que compoñen o ciclo de vida do proxecto en pequenas etapas repetitivas chamadas iteracións. O obxetivo principal desta metodoloxía é que, ao acabar cada iteración se teña unha versión funcional do producto, a cal cumpra cun conxunto de requisitos definidos na fase de análise. Desta forma, cada iteración partirá da anterior e engadirá novas funcionalidades ao producto ata chegar á versión final do mesmo, na cal se deben de cumplir todos os requisitos establecidos na fase de análise.

O principal motivo polo que se escolleu esta metodoloxía frente a outras máis tradicionais foi o importante número de vantaxes que presenta, entre as que destacan:

- Permite separar a complexidade do proxecto ao dividilo en iteracións.
- O cliente que encargue a aplicación non ten que esperar ata o final do seu desenvolvemento para probala xa que, en cada iteración, se proporciona unha versión funcional da mesma. Esta característica da metodoloxía otorga a posibilidade de que o cliente vaia probando a aplicación para dar un *feedback* aos desarrolladores, facendo que estes poidan adaptar a aplicación máis rapidamente ás súas esixencias.
- A detección de erros é moito máis doada, o que conleva que sexa máis difícil que estes erros se propaguen a través das iteracións e cheguen a versión final do produto.

- Adquírese experiencia dos requerimentos dos incrementos posteriores do sistema a través da utilización das primeiras iteracións como prototipos.

4.2 Recursos

Á hora de realizar un proxecto, unha das partes máis importantes é determinar os recuros humanos e materiais (hardware, ferramentas software, etc) que serán necesarios para que se poida levar a cabo. Estes recursos teñen un custo económico asociado, polo que a viabilidade económica do proxecto depende, en gran medida, da planificación realizada e da estimación de recursos necesarios para o seu desenvolvemento. Neste proxecto, precísanse 3 tipos de recursos:

4.2.1 Recursos hardware

Os recursos hardware son aqueles dispositivos que son necesarios para a realización do proxecto. Neste proxecto, necesitouse facer uso do ordenador portátil do autor do traballo, un dispositivo BCI proporcionado polo Grupo de Tecnoloxía Electónica e Comunicacións (GTEC) e unha cámara para realizar a gravación dos vídeos de proba.

4.2.2 Recursos software

Os recursos software consisten no conxunto de ferramentas software que son necesarias para a realización do proxecto. Todas as ferramentas que foi necesario empregar foron xa presentadas no capítulo 3 desta memoria.

4.2.3 Recursos humanos

Conxunto de persoas que son necesarias para a realización do proxecto ou que interveñen de forma directa no se desenvolvemento. Neste proxecto, estableceuse un grupo de traballo formado por 3 persoas, as cales desempeñaron os seguintes roles:

- **Director:** Rol desempeñado por Óscar Fresnedo Arias e Francisco Laport López. É o encargado de supervisar o traballo realizado polo analista e o desarrollador.
- **Analista:** Rol desempeñado polo autor do proxecto. O seu cometido consiste en determinar o requisitos funcionais e non funcionais da aplicación.
- **Desarrollador:** Rol desempeñado polo autor do proxecto. O seu cometido consiste na realización do diseño, da implementación e das probas da aplicación que se ten que levar a cabo, cumprindo os requisitos establecidos polo analista e seguindo as indicacións do director.

4.3 Planificación

Nesta sección, vaise a explicar a planificación inicial que se levou a cabo seguindo a metodoloxía iterativa incremental. A realización do proxecto foi dividida nas seguintes iteracións:

- **Iteración 1 :** Análise e determinación dos requisitos funcionais e non funcionais da aplicación.
- **Iteración 2 :** Análise das ferramentas e tecnoloxías necesarias para a realización do proxecto.
- **Iteración 3 :** Implementación do módulo de detección básica de pestanexos a partir das encefalografías capturadas polo BCI.
- **Iteración 4 :** Implementación do módulo de recoñecemento de obxectos con OpenCV.
- **Iteración 5 :** Implemetación da lóxica necesaria no nodo central para integrar a información recibida e tomar as decisións correspondentes en función dos mesmos.
- **Iteración 6 :** Configuración da comunicación entre os dispositivos presentes no proxecto mediante a utilización do protocolo de comunicacións MQTT.

Compre resaltar que, na iteración 3, os módulos que serán en implementados nas iteracións 4 e 5 devolverán uns valores fixos e a comunicación será local, é decir, terán un funcionamento simulado para poder probar a funcionalidade que se está a implementar. A medida que se vaia avanzando nas iteracións, estas simulacións serán substituídas pola implementación final do módulo correspondente que terá a aplicación unha vez rematada.

Nas figuras 4.1 e 4.2, móstrase a planificación inicial establecida e o diagrama de Gantt asociado, respectivamente. Na táboa de planificación, poden apreciarse as iteracións que compoñen o proxecto divididas nas súas correspondentes tarefas, indicando a súa duración e os recursos humanos involucrados en cada unha delas. Como se pode apreciar, a duración de cada iteración depende directamente da complexidade das tarefas que a conforman e da dispoñibilidade dos recursos existentes. Debido as limitacións do autor, estableceuse unha xornada laboral de 3 horas.

4.4 Análise económico

Unha vez feita a planifiación do proxecto, decidíuse realizar unha análise económica mediante a ferramenta Project Libre[30] para ter unha estimación realista dos custos que require o desenvolvemento do proxecto. O custo total do proxecto será a suma do soldo asociado ao

Nombre	Duracion	Trabajo	Inicio	Terminado	Nombres del Recurso	Predecesores	Costo
Análise de requisitos	13 days	45 horas	**25/06/20 8:00**	**1/07/20 16:00**			1287,00 €
Estudo dos módulos necesarios	7 days	21 horas	25/06/20 8:00	29/06/20 14:00	Analista		546,00 €
Especificar obxectivos do proxecto e requisitos	5 days	15 horas	29/06/20 14:00	1/07/20 13:00	Analista	2	390,00 €
Revisión do análise	1 day	9 horas	1/07/20 13:00	1/07/20 16:00	Analista;Director 1;Director 2	3	351,00 €
Análise das tecnoloxías e ferramentas	18 days	60 horas	**1/07/20 16:00**	**10/07/20 14:00**			1677,00 €
Estudo das posibles tecnoloxías e ferramentas a utilizar	12 days	36 horas	1/07/20 16:00	8/07/20 11:00	Analista	4	936,00 €
Elección das tecnoloxías e ferramentas	5 days	15 horas	8/07/20 11:00	10/07/20 10:00	Analista	6	390,00 €
Revisión das eleccións por parte dos directores	1 day	9 horas	10/07/20 10:00	10/07/20 14:00	Analista;Director 1;Director 2	7	351,00 €
Implementación do módulo de detección de pestanexos	23 days	75 horas	**10/07/20 14:00**	**23/07/20 10:00**			1723,80 €
Implementación da funcionalidade simulada	3 days	9 horas	10/07/20 14:00	13/07/20 15:00	Desarrollador	8	187,20 €
Implementación do filtro paso banda	5 days	15 horas	13/07/20 15:00	15/07/20 14:00	Desarrollador	10	312,00 €
Cálculo dos umbrais e o intervalo de pestanexo	7 days	21 horas	15/07/20 14:00	20/07/20 10:00	Desarrollador	11	436,80 €
Implementación do detector de pestanexos	7 days	21 horas	20/07/20 10:00	22/07/20 16:00	Desarrollador	12	436,80 €
Revisión do módulo de detección de pestanexos	1 day	9 horas	22/07/20 16:00	23/07/20 10:00	Analista;Director 1;Director 2	13	351,00 €
Implementación do módulo de detección de obxectos	25 days	81 horas	**23/07/20 10:00**	**5/08/20 14:00**			1848,60 €
Carga do módelo preentrenado YOLOv3	2 days	6 horas	23/07/20 10:00	23/07/20 17:00	Desarrollador	14	124,80 €
Implementación do análise e elección dos frames	10 days	30 horas	24/07/20 8:00	29/07/20 15:00	Desarrollador	16	624,00 €
Implementación da detección e recoñecemento de obxectos	12 days	36 horas	29/07/20 15:00	5/08/20 10:00	Desarrollador	17	748,80 €
Revisión do módulo de detección de obxectos	1 day	9 horas	5/08/20 10:00	5/08/20 14:00	Analista;Director 1;Director 2	18	351,00 €
Implementación do módulo de procesamento central	13 days	45 horas	**5/08/20 14:00**	**12/08/20 13:00**			1099,80 €
Integración dos módulos de detección de obxectos e pestanexos	5 days	15 horas	5/08/20 14:00	7/08/20 13:00	Desarrollador	19	312,00 €
Implementación da elección de ordes a enviar	7 days	21 horas	7/08/20 13:00	12/08/20 9:00	Desarrollador	21	436,80 €
Revisión do módulo de procesamento central	1 day	9 horas	12/08/20 9:00	12/08/20 13:00	Analista;Director 1;Director 2	22	351,00 €
Implementación do módulo de conectividade	19 days	63 horas	**12/08/20 13:00**	**21/08/20 14:00**			1474,20 €
Implementación da recepción de datos	9 days	27 horas	12/08/20 13:00	17/08/20 16:00	Desarrollador	23	561,60 €
Implementación do envío de datos	9 days	27 horas	17/08/20 16:00	21/08/20 10:00	Desarrollador	25	561,60 €
Revisión final da aplicación	1 day	9 horas	21/08/20 10:00	21/08/20 14:00	Analista;Director 1;Director 2	26	351,00 €

Figura 4.1: Planificación das iteracións.

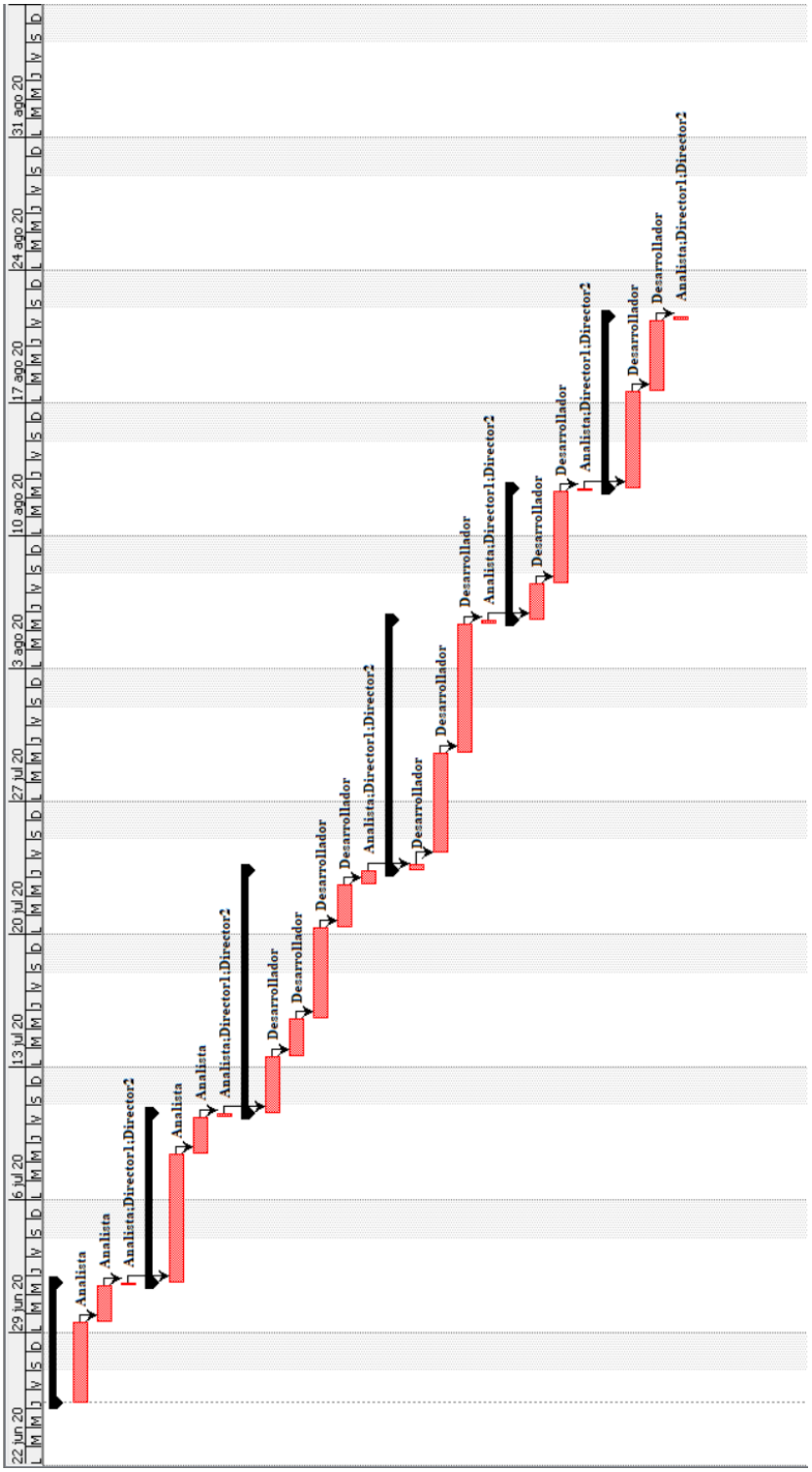

Figura 4.2: Diagrama de Gantt.

diferentes recursos humanos que interveñen nel e dos custos dos recuros materiais. Tendo en conta que os recursos hardware xa estaban dispoñibles no momento de comezar o proxecto e que o software utilizado non supuxo ningún custo a maiores, todo o gasto no proxecto procede dos recursos humanos. Na táboa 4.1, móstranse o soldo por hora de traballo que conleva cada recurso humano, os cales foron estimados a partir do informe *Guía Salarial Sector TI Galicia 2015-2016* [31]. Tendo en conta estos datos e o número de horas asignadas a cada traballador na planificación, o custo total do proxecto estimouse en 9110,40 €.

Recursos humanos	Director	Analista	Desarrollador
Óscar Fresnedo Arias	45,50 €	-	-
Francisco Laport López	45,50 €	-	-
Carlos Varela Manteiga	-	26,00 €	20,80 €

Táboa 4.1: Custos dos recursos humanos.

O resumo da información máis relevante do proxecto móstrase na táboa 4.2.

Data de inicio	Data de finalización	Duración	Traballo	Custo
25/06/20	21/08/20	111 días	369 horas	9110,40 €

Táboa 4.2: Información xeral do proxecto.

4.5 Seguemento do proxecto

A planificación que se expuxo na sección anterior é unha planificación inicial que se cumpriría no caso de que todas as iteracións se realizasen exitosamente dentro dos prazos establecidos. No caso deste proxecto, a planificación incial non se puido cumprir perfectamente debido a que se producíu un retraso de 3 días na tarefa de implementación da detección e recoñecementos de obxectos, que está presente na cuarta iteración. O motivo polo que se producíu este retraso foi que houbo que realizar varios cambios na implementación que se fixo inicialmente, xa que esta proporcionaba varias detección erróneas.

Finalmente hai que deixar constancia de que este retraso conlevou un aumento de 187,20 € no custo total do proxecto, polo que este pasará a ter un custo real de 9297,60 €.

Capítulo 5

Visión xeral do proxecto

O primeiro paso que se realizou na elaboración deste proxecto foi pensar nunha visión xeral do mesmo, xa que servirá de guía para comprender as diferentes partes que o compoñen, a comunicación existente entre as mesmas e os pasos a seguir para levalo a cabo. Neste capítulo vaise a presentar a arquitectura que se deseñou para a aplicación cos diferentes módulos que a compoñen, xunto cunha explicación das diferentes operacións que é necesario levar a cabo para cumplir cos requisitos establecidos.

5.1 Análise de requisitos

Nun proxecto software, un dos pasos iniciais máis importantes é a análise e definición dos requisitos do sistema. Estes requisitos establecen os servicios que teñen que ser proporcionados polo producto a desenvolver, así como calquera tipo de restricción. Debido a que neste proxecto se vai a facer uso dunha metodoloxía iterativa incremental, este proceso será levado a cabo ao longo do desenvolvemento do proxecto xa que, a maiores da realización de entrevistas entre o cliente e o desarrollador ao inicio do proxecto, os desarrolladores teñen a posibilidade de que os clientes poidan probar o producto ao final das fases ou iteracións nas que se divide o desenvolvemento do proxecto.

Os requisitos dun sistema pódense dividir en 2 tipos: requisitos funcionais e requisitos non funcionais.

5.1.1 Requisitos funcionais

Os requisitos funcionais consisten na declaración dos servicios que proveerá o produto a desenvolver, na exposición do comportamento do producto ante unhas entradas particulares e na declaración explícita do que o sistema non debe de facer [32]. Na maioría dos casos, unha gran parte dos problemas que aparecen durante desenvolvemento dun produto software

son consecuencia dunha especificación pobre ou pouco detallada deste tipo de requisitos, polo que se considera un proceso crítico e de vital importancia.

Para a realización da nosa aplicación definíronse os seguinte requisitos funcionais:

- **Formatos dos datos de entrada:** Os arquivos enviados por parte do BCI teñen que ter un formato .txt e os arquivos enviados pola cámara un formato .avi ou .mp4.
- **Solución de interconexión dos dispositivos:** A aplicación debe de implementar todas as operacións necesarias para a interconexión dos dispositivos que se van a ver involucrados no proxecto. Para iso farase uso do protocolo de comunicación MQTT e da librería de Python paho-mqtt, a cal nos permitirá implementar as operacións do protocolo que vaian a ser utilizadas.
- **Detección de obxectos:** A aplicación debe permitir detectar obxectos cotiás que estén presentes no entorno do usuario. No caso do noso proxecto o detector está adestrado para detectar unha serie de obxectos cotiás que forman parte do *dataset* COCO [19].
- **Definir varias accións:** A aplicación debe de implementar a posibilidade de definir varias accións para un mesmo obxecto a través de diferentes secuencias de pestanexos.
- **Obxectos de combinación:** Co obxectivo de evitar falsas detección con obxectos como televisións, os cales se observan fixamente, a aplicación ten que implementar a posibilidade de ter que detectar dous obxectos concretos para controlar un. Por exemplo, ter que detectar unha televisión e o mando da mesma para poder controlar a televisión.
- **Detección de pestanexos:** A aplicación debe implementar un detector de pestanexos que sexa capaz de detectar o número de pestanexos existentes nunha intervalo de tempo do EEG a partir dos arquivos .txt que son enviados polo BCI.
- **Nodo de procesamento central:** A aplicación debe de implementar un nodo de procesamento central, o cal se encargará de determinar a orde que se quere enviar a un dispositivo destino en función do obxecto e o número de pestanexos detectados.

5.1.2 Requisitos non funcionais:

Son aqueles requisitos que non se refiren directamente as funcionalidades específicas que integra o produto, senón a outras propiedades desexables de éste como son a fiabilidade, a resposta no tempo ou a capacidade de almacenamento [32].

Para a realización da nosa aplicación definíronse os seguintes requisitos non funcionais:

- **Configuración do entorno:** A aplicación ten que ser facilmente integrable en diferentes tipos de dispositivos, evitando a necisade de realizar instalacións de paquetes ou programas largos e complexos.
- **Usabilidade:** A aplicación ten que ser sinxela de utilizar e os dispositivos que son necesarios para o funcionamento da mesma teñen que ser cómodos de utilizar por parte dos usuarios.
- **Fiabilidade:** A aplicación debe de proporcionar resultados correctos e fiables para calquera combinación de datos de entrada que estea contemplada na implementación do dispositivo de procesamento central.
- **Mantemento:** O software ten que ser facilmente adaptable e modificable para engadir novas funcionalidades e adaptalo á situación de cada usuario.
- **Tempo de resposta:** A aplicación ten que conseguir executar a orde que queira levar a cabo o usuario nun período de tempo reducido, e decír, non pode ter tempos de resposta moi elevados.

5.2 Arquitectura do sistema

Nesta sección vaise explicar a arquitectura creada para o sistema de dispositivos dos que fará uso a aplicación a desenvolver neste proxecto. Na figura 5.1, podemos ver unha ilustración da arquitectura en cuestión.

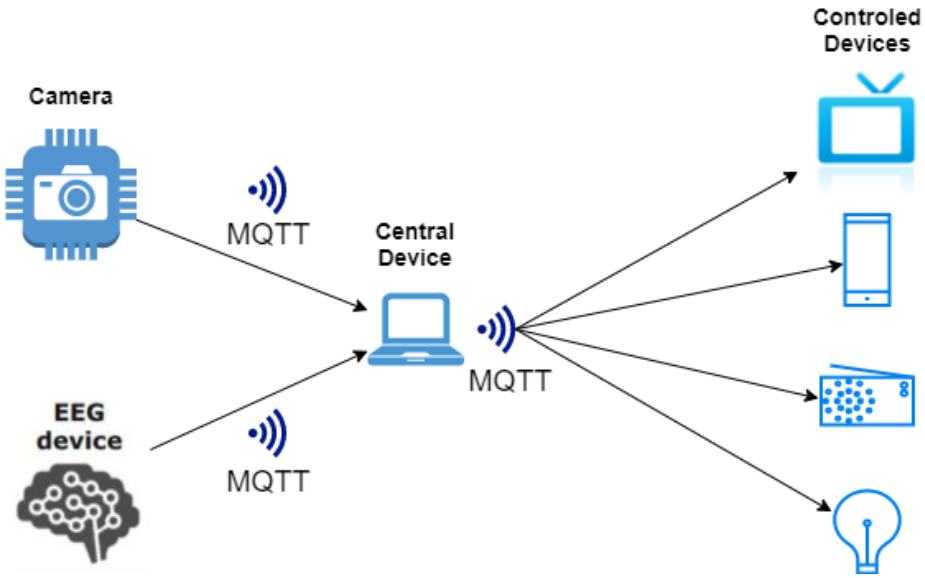

Figura 5.1: Arquitectura do sistema.

Tal e como se pode apreciar, a aplicación segue un deseño similar ao de un sistema clásico de IoT, donde temos unha serie de sensores encargados de recoller a información de interese, un nodo central que se encarga de procesar esa información e tomar unha decisión, e os nodos actuadores que executan as accións enviadas polo nodo central. En concreto, o sistema está formado por estos catro tipos de dispositivos:

- **Cámara:** Este dispositivo, ao que na figura se fai referencia como "*Camera*", será o encargado de gravar o que vé o usuario en todo momento. Esta secuencia de frames que é recopilada pola cámara será enviada ao nodo central para o seu procesamento.
- **BCI:** Este dispositivo, ao que na figura se fai referencia como "*EEG Device*", será o encargado de recopilar a actividade cerebral do usuario e almacenar as correspondentes mostras do sinal en arquivos .txt para o seu posterior envío ao dispositivo de procesamento central.
- **Dispositivo de procesamento central:** Referenciado na figura como "*Central Device*", este dispositivo será o encargado de procesar a información enviada por parte da cámara e o BCI para determinar que dispositivo quere controlar o usuario e que acción quere levar a cabo sobre el. Unha vez se determine a acción, este dispositivo enivará un comando ao dispositivo final que se queira controlar. Ademáis, desempeñara o papel de broker MQTT.
- **Dispositivo a controlar:** Referenciados na figura como "*Controled Device*", son os dispositivos que poden ser controlados por parte do usuario a través da aplicación. Este tipo de dispositivos simplemente recibirán e executarán os comandos enviados por parte do dispositivo de procesamento central.

Unha vez sabemos o papel que desempeña cada un dos dispositivos que forman o sistema hai que resaltar as comunicacións existentes entre os dispositivos. Na figura 5.1 pódese observar que a cámara e o BCI están conectados co dispositivo de procesamento central e que este último está conectado con todos os dispositivos que poidan ser controlados por parte da aplicación. Estas conexións son creadas mediante a utilización do protocolo de comunicación inalámbrica MQTT, o cal permitirá que a cámara e o BCI poidan enviar os seus arquivos ao dispositivo de porcesamento central e que este último envíe unha mensaxe co comando a executar aos dispositivos que o usuario queira controlar.

5.3 Diagrama de fluxo

Para ilustrar graficamente o funcionamento xeral da aplicación, creouse o diagrama de fluxo que se mostra na figura 5.2. Neste diagrama detállanse cada unha das operacións que é preciso

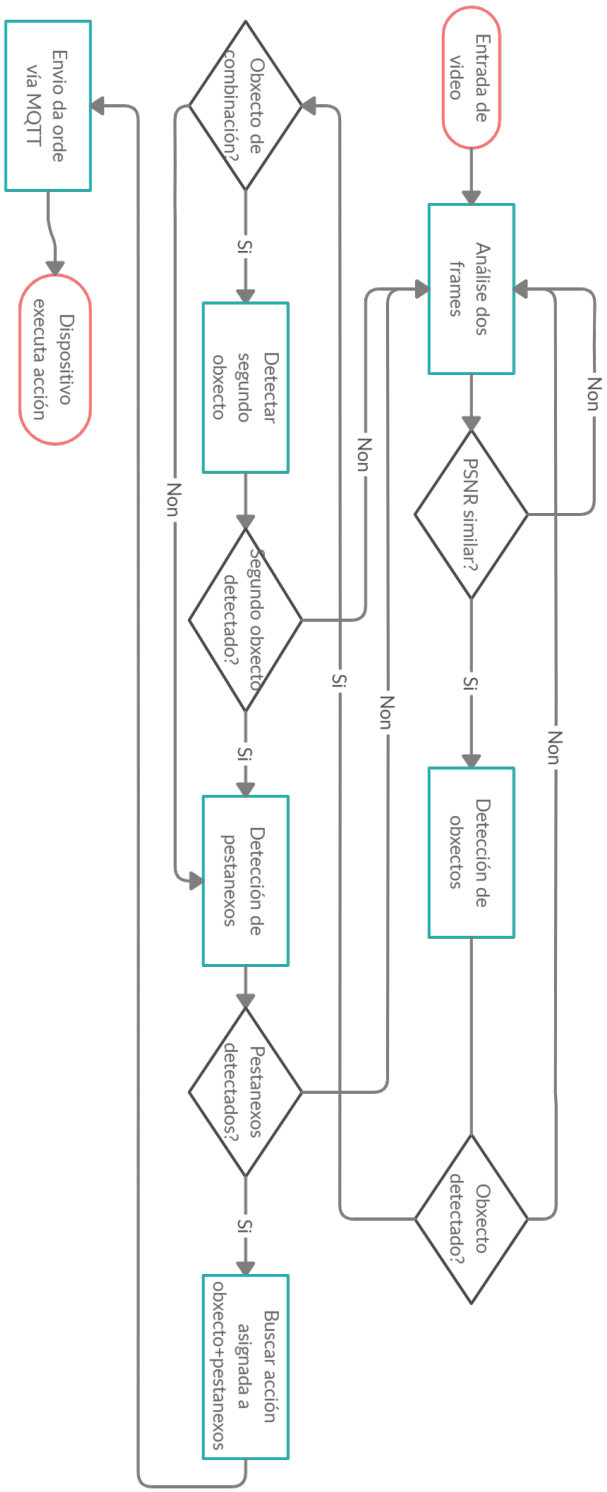

Figura 5.2: Diagrama de fluxo da aplicación.

realizar a partir dos datos de entrada, os eventos que desencadean o inicio de cada unha delas e as diferentes transicións que ocorren ata enviar o sinal de activación ou orde de control ao obxecto/dispositivo correspondente.

Tal e como se pode apreciar no diagrama da figura 5.2, as principais operacións da aplicación son as seguintes:

1. **Procesamento dos *frames*:** O vídeo captado pola cámara do noso dispositvo é enviado ao dispositivo central para o seu procesamento. En primeiro lugar, os *frames* que compoñen o vídeo son seleccionados, escollendo só unha fracción deles para o seu procesamento. A continuación, faise unha avaliación da similitude que hai entre cinco *frames* consecutivos usando a métrica do PSNR, xa que esta métrica nos indica se existe moita variación ou non entre esos cinco *frames* consecutivos. Se non existe moita variación, o último dos *frames* avaliados é procesado e, en caso contrario, o primeiro *frame* será descartado e pasarase ó seguinte. Isto débese a que unha das condicións que se establecen para a detección de obxectos é que o usuario ten que miralo fixamente, polo que se o PSNR mostra moita variación o primeiro *frame* é descartado, xa que normalmente esta variación é debida a que a persoa non está mirando nada en concreto, polo que non ten sentido proceder coa detección.

   Compre aclarar que neste paso seleccionouse unha fracción dos *frames* que compoñen o vídeo debido a que son suficientes para levar a cabo a operación de detección de obxectos sen empeorar as taxas de detección e que, tanto esta medida como o filtrado usando a métrica do PSNR, foron implementadas co obxectivo de reducir o custo computacional da operación da detección de obxectos e aumentar a velocidade de execución do detector.

2. **Detección de obxectos:** O seguinte paso consistirá en chamar ao detector de obxectos para que intente recoñecer algún dos obxectos de interese no *frame* que se lle pasou. Se non se detecta ningún, é dicir, se o *frame* non contén ningún obxecto de interese polo detector, o *frame* pasa a descartarse e cóllese o seguinte. Se pola contra se detecta un obxecto, o seguinte paso consistirá en comprobar se este é un obxecto de combinación.

3. **Obxectos de combinación:** No noso proxecto contémplase a posiblidade de que, para controlar un dispositivo concreto, se teña que detectar unha combinación de obxectos como, por exemplo, o mando + TV para encender a televisión. Polo tanto, cando se detecta un obxecto hai que comprobar se é necesario detectar outro, é dicir, se se trata dun obxecto de combinación. En caso de non ser necesario, o paso "Detectar segundo obxecto" omítese e pásase directamente á detección de pestanexos. Se, pola contra,

sí é necesario, intentarase detectar o segundo obxecto, collendo como *frame* inicial o seguinte ao *frame* no que se detectou o primeiro obxecto e establecendo unha cantidade límite de frames a procesar. Se o obxecto de combinación non é detectado antes de rematar o límite de frames establecido ou antes de rematar os frames do vídeo que faltaban por procesar, o obxecto que se detectou inicialmente será descartado e volverase ao primeiro paso do fluxo, collendo como *frame* inicial o seguinte ao que se estaba a procesar. No caso de que o segundo obxecto sexa detectado, pasarase ao paso de "Detección de pestanexos".

4. **Detección de pestanexos:** Tras detectar o/os obxecto/s de interés, temos que determinar a secuencia de pestanexos que realiza o usuario, xa que esta combinación fará que o dispositivo central determine a acción que se ten que executar sobre o obxecto detectado. O detector de pestanexos collerá como entrada os datos enviados polo BCI, os cales permiten obter o sinal de EEG a partir do cal obteremos o número de pestanexos que o usuario realizou despois de fixar a mirada no obxecto de interese. Unha vez detectado o obxecto, establécese unha xanela temporal na que se procede coa detección dos pestanexos. Se os pestanexos son detectados e a secuencia coincide con algunha acción prestablecida sobre o obxecto de interese séguese co seguinte paso pero, en caso de non detectar ningún ou dunha secuencia non recoñecida, descártaranse todos os pasos feitos anteriormente e analizarase o seguinte *frame*.

5. **Búsqueda da acción a realizar:** Tras obter o obxecto detectado e o número de pestanexos, o dispositivo de porcesamento central será o encargado de determinar a acción que se levará a s sobre o dispositivo que se quere controlar en función destes dous parámetros.

6. **Envío da orde e execución da acción:** O derradeiro paso consiste en enviar a orde correspondente ao obxecto sobre o que se quere realizar a acción. O dispositivo central enviará esta orde mediante o protocolo de comunicación sen fíos MQTT, facendo que o obxecto a controlar execute a acción que se indicaba na mesma.

Capítulo 6

Detección de obxectos

Tal e como se comentou na introdución, a aplicación permitará aos usuarios controlar dispositivos do seu entorno tan só co pestanexo dos seus ollos, así que, para levar a cabo accións sobre diferentes dispositivos, necesitamos un módulo que nos permita detectalos e recoñecelos de maneira efectiva. O detector de obxectos que se explica neste capítulo foi creado con ese fin, detectar e recoñecer dispositivos da contorna do usuario que poidan ser controlados.

6.1 Datos de entrada

O detector de obxectos recibirá como datos de entrada os *frames* corrrespondentes a un vídeo en formato .avi ou .mp4, que será enviado ao nodo de procesamento central, onde se executa o detector, por parte dun dispositivo que teña unha cámara incorporada. Este dispositivo pode ser unha simple minicámara ou unha raspberrypi cunha pequena cámara incoporada, a cal se encontrará integrada no BCI que se lle colocará ao usuario. As gravacións realizadas pola cámara serán enviadas constantemente ao dispositivo de procesamento central mediante o protocolo de comunicación MQTT, polo que a cámara utilizada ten que contar cun chip ou cun adaptador Wi-Fi que lle permita realizar comunicacións a través deste protocolo.

6.2 Procesamento dos datos

A medida que o detector vai recibindo os *frames* do vídeo como entrada, o seguinte paso é procesar esos *frames* para averiguar se o usuario está intentando realizar algunha acción sobre un obxecto que poida controlar. Este procesamento divídese nos seguinte pasos:

1. **Carga de arquivos de configuración:** O primeiro paso consiste en cargar os arquivos de configuración necesarios para o funcionamento do detector de obxetos. En primeiro lugar cárgase o arquivo coco.names, o cal contén as etiquetas/*labels* dos obxetos que formaban parte do *dataset* co que foi adestrado o noso detector, que neste caso é o

dataset COCO [19]. Ademais dos *labels*, necesítanse cargar os arquivos yolov3.weights e yolov3.cfg, os cales son necesarios para cargar o modelo preadestrado de YOLOv3 que se vai a utilizar como detector [33].

2. **Limitación dos frames:** A idea inicial que se tiña para procesar o vídeo de entrada era analizalo *frame* a *frame* para determinar se se detectaba algún obxecto, pero esta estratexia foi descartada debido aos problemas de rendemento que presentaba e a carga computacional que supoñía. Tras varias probas, chegouse a conclusión de que, procesando só unha parte dos *frames*, o detector conseguía recoñecer igualmente os obxectos de interese de maneira eficaz e cun custo computacional moi inferior. Tendo en conta a taxa de *frames* por segundo dos vídeos actuais e as probas que se realizaron, decidiuse que o noso detector só analizase 1 de cada 10 frames, reducindo notablemente o tempo de procesamento e a carga computacional, e conservando a eficacia e fiabilidade que presentaba anteriormente. Isto débese a que as cámaras actuais utilizan moitos FPS nas súas gravacións, sendo esta cantiadade tan grande de *frames* totalmente innecesaria para o noso obxetivo, xa que a maioría contén información redundante.

3. **Omisión de intervalos irrelevantes:** Seguindo o diagrama de fluxo que se diseñou para entender a lóxica da aplicación (figura 5.2), o usuario ten que fixar a súa mirada no dispositivo que queira controlar e, posteriormente, realizar unha serie de pestanexos. En función disto, chegouse a conclusión de que se se detecta moita variación entre varios frames consecutivos non tería sentido que estes fosen analizados polo noso detector, xa que esta variación conlevaría que o usuario non está fixando a súa mirada en ningún dispositivo concreto e, polo tanto, non está intentando controlar ningún. Tras esta reflexión, decidiuse que o detector evitase procesar este tipo de frames, xa que con isto melloraríase considerablemente o rendemento e o tempo de resposta da aplicación.

   Para evitar ter que chamar ao detector neste tipo de situacións, implementouse un sinxelo mecanismo para comprobar se varios *frames* consecutivos son similares usando como métrica o PSNR. Esta métrica permite medir o grado de similitude existente entre dous frames tomando o primeiro *frame* como referencia e comparándoo co seguinte. Polo tanto, de xeito indirecto, tamén nos indica se dous frames son significativamente diferentes e non comparten unha estrutura común. No caso da nosa aplicación, cada vez que chega un novo *frame* para procesar, calcúlase o valor de PSNR respecto do seu *frame* anterior. Se ese valor supera un certo umbral indica que son suficientemente similares e repítese o mesmo proceso ata que se procesan 10 frames consecutivos que cumplan esa condición. No momento no que un dos *frames* non cumpla a condición do umbral de PSNR, comézase de novo ata obter 10 *frames* que sexan suficientemente similares. Se os valores proporcionados pola métrica PSNR indican que existe pouca

variación entre 10 *frames* consecutivos, o último *frame* deste conxunto será enviado ao algortimo de detección.

4. **Omisión de intervalos de inactividade:** Unha vez se obteña un *frame* a procesar mediante o mecanismo de comprobación anterior, este será enviado ao detector baseado en YOLO para que o procese e se obteñan o/os obxecto/s de interese que estén presentes en dito *frame*. No caso de que o detector non recoñeza ningún obxecto, collerase o seguinte *frame* e compararase mediante a métrica do PSNR co *frame* que utilizou o detector. Se estes *frames* resultan moi similares en termos de PSNR, o novo *frame* será descartado e repetirase este proceso cos sucesivos *frames* ata que se reciba un o suficientemente diferente con respecto ao *frame* utilizado polo detector. O motivo polo que se leva a cabo este proceso é a necesidade de evitar chamar ao detector cando se sabe con seguridade que este non detectará nada. Resulta lóxico pensar que, se non se recoñeceu ningún obxecto de interese e os seguintes *frames* son moi parecidos (indicativo de que o suxeito segue fixando a mirada no mesmo punto), o detector tampouco detectará ningún obxecto nesos *frames*. Neste caso, a métrica de PSNR emprégase para buscar *frames* con diferente estructura, polo que se pode empregar o mesmo umbral que no caso anterior pero co próposito contrario.

5. **Obtención dos labels/identificadores dos obxectos detectados:** Se o detector YOLO detecta obxectos de interés no *frame* que está a procesar, este devolverá os identificadores ou *labels* correspondentes a ditos obxectos e unha taxa de fiabilidade. Este valor indícanos o nivel de certeza que temos sobre a detección dun obxecto. No caso de que se detecten varios obxectos de interese no *frame*, estes serán ordenados para o seu análise en función da súa taxa de fiabilidade, dando prioridade ao que teña unha taxa máis elevada. Tras isto, comprobarase que a taxa de fiabilidade do obxecto seleccionado supera un valor mínimo que é prefixado de antemán. Se un obxecto alcanza ou supera ese valor mínimo, o *label* asociado ao mesmo será gardado polo nodo de procesamento central. Se ningún dos obxectos detectados superase ou alcanzase o valor mínimo de fiabilidade, asúmese que a detección non tivo éxito e pásase a procesar o seguinte *frame*.

6.3 Decisións de deseño

Nesta sección vanse a aclarar e xustificar as decisións de deseño que se levaron a cabo durante o desenvolvemento deste módulo. Estas decisións de deseño están directamente relacionadas con definir o valor dos diferentes parámetros do sistema que determinarán o comportamento deste módulo de detección.

6.3.1 Fracción de *frames* seleccionados do *streaming* de vídeo

Tal e como se expuxo neste capítulo, o módulo de detección de obxectos só selecciona unha fracción dos *frames* totais que compoñen o vídeo a procesar. Para determinar a fracción de *frames* que se van procesar, realizáronse varias execucións do detector para comprobar que tempos de execución e que taxa de erro na detección se obtiñan considerando diferentes fraccións de *frames*. Tras a realización destas probas chegouse á conclusión de que, utilizando só 1 de cada 10 *frames*, o tempo de execución do detector reducíase considerablemente ata un punto razoable, mantendo unha taxa de erro similar á que se obtiña procesando todos os *frames*. Isto débese fundamentalmente a que as cámaras actuais realizan gravacións cunha cantidade de *frames* moi elevada, polo que a maioría destes *frames* representan información redundante para o noso detector.

6.3.2 Número de *frames* consecutivos similares para iniciar detección

Para determinar se o usuario ten a mirada fixa nun obxecto que quere detectar, é razoable pensar que varios *frames* consecutivos do vídeo teñen que ser similares, xa que isto é indicativo de que a persoa fixou a súa mirada. Nesta caso, decidiuse que este número de frames consecutivos sexa 5. Unha boa parte das cámaras actuais gravan vídeo entorno a 25 fps ou 30 fps. Esto quere dicir que cada segundo de vídeo vai a ser representado por 25 ou 30 *frames*. Partindo de que só se selecciona 1 de cada 10 *frames*, establecer este parámetro a 5 supón considerar aproximadamente 2 segundos dun vídeo gravado con este tipo de cámaras. De todas formas, este valor pode configurarse se a taxa de gravación é maior para que o intervalo de tempo considerado se achegue a esos 2 segundos igualmente.
Polo tanto, fixando este parámetro a ese valor implica que o usuario estivo mirando un determinado punto durante 2 segundos completos, o que para nós é un intervalo de tempo sufucientemente longo como para pensar que o usuario está fixando a súa mirada nun punto deliveradamente co obxectivo de realizar unha acción sobre un obxecto.

6.3.3 Umbral do PSNR

Neste módulo faise uso da métrica do PSNR para analizar se existe ou non moita variación entre *frames*. Debido a súa formulación matématica (ver 2.1) e á natureza das imaxes habituais, esta métrica proporciona valores por debaixo dos 10 dB cando dous *frames* son significativamente diferentes, valores por enriba dos 40 dB cando dous *frames* presentan moi pouca variación e valores entre 20 e 30 dB cando dous *frames* son relativamente parecidos. Estes rangos de valores son orientativos pero, despois de décadas de investigación en procesado de imaxes e fotografías, son bastante fiables e axustados á realiade. Tendo en conta isto, no proxecto decidiuse establer o umbral de PSNR empregado a un valor intermedio. En concreto, o

umbral de PSNR estableceuse a 20 dB. Deste xeito, para analizar se dous *frames* son relativamente parecidos podemos comprobar se o valor devolto polo PSNR é igual ou superior a 20 e, por outra parte, comprobar se é inferior a 20 para analizar se dous *frames* son relativamente diferentes entre si.

6.3.4 Umbral da taxa de fiabilidade

O valor do umbral da taxa de fiabilidade foi establecido a 0.75. Este valor estableceuse mediante a realización de varias execucións do módulo con valores diferentes para analizar cantas falsas deteccióǹs se producen. Tras realizar estas probas, puídose comprobar que a partir deste valor suprímense practicamente todas as falsas deteccióǹs.

6.4 Probas do módulo

Nesta sección vanse a expoñer as diferentes probas que se levaron a cabo para comprobar o correcto funcionamento do módulo, verificando que se cumpran os diferentes requisitos que foron definidos ao inicio do proxecto.

6.4.1 Avaliación do número de *frames* seleccionados

Para determinar a fracción de *frames* do vídeo que se van seleccionar, levouse a cabo unha análise dos tempos de execución arroxados polo módulo de detección en función da cantidade de *frames* seleccionados e das seguintes taxas de detección:

- **Falsos Positivos:** Número de deteccióǹs de obxectos nas que o detector devolve un obxecto equivocado.
- **Falsos Negativos:** Número de veces nas que o detector devolve que non se detectou ningún obxecto e, sen embrago, no vídeo de entrada sí que existen obxectos que este debería detectar.
- **Verdadeiros Positivos:** Número de deteccióǹs nas que o detector leva a cabo a detección dun obxecto con éxito.
- **Verdadeiros Negativos:** Número de veces nas que o detector devolve que non se detectou ningún obxecto e, efectivamente, non existe ningún obxecto que detectar no vídeo de entrada.

Para esta proba, preparáronse 45 vídeos con diferentes características de 10 segundos de duración cada un deles e simulando posibles situacións reais. Destes 45 vídeos, 30 deles contiñan

diferentes obxectos de interese que o detector debería de detectar no caso de que se cumpran as condicións impostas no módulo e os restantes contiñan outro tipo de obxectos. Estes vídeos foron procesados polo módulo de detección de obxectos considerando 4 valores diferentes para a fracción de *frames* que se queren seleccionar: todos os *frames* existentes, 1 de cada 5, 1 de cada 10 e 1 de cada 20. A partir das diferentes execucións, obtivéronse os datos que se mostran na táboa 6.1.

Número de *frames*	Tempo de execución	Falsos Positivos	Falsos Negativos	Verdadeiros Positivos	Verdadeiros Negativos
Todos	190,82 segundos	4	1	29	11
1 de cada 5	40,32 segundos	2	3	27	13
1 de cada 10	22,33 segundos	1	3	27	14
1 de cada 20	14,54 segundos	1	8	22	14

Táboa 6.1: Datos obtidos a partir das execucións realizadas.

Tras a obtención dos datos, estes foron avaliados utilizando a métrica de *Recall* ou de exhaustividade para obter a porcentaxe de obxectos de interese que foron detectados correctamente. Para calcular o valor desta métrica utilizarase a fórmula: Recall = TP/(TP + FN), onde TP son os verdadeiros positivos (número de obxectos existentes nos vídeos que foron detectados correctamente) e FN son os falsos negativos (número de obxectos existentes que non foron detectados cando deberían ter sido detectados). Polo tanto, no noso caso obteránse os seguintes valores:

Número de *frames*	Valor da métrica
Todos	0,96
1 de cada 5	0,9
1 de cada 10	0,9
1 de cada 20	0,73

Táboa 6.2: Resultados da métrica *Recall*.

Analizando detidamente os datos obtidos no experimento, chegouse a conclusión de que a mellor opción era seleccionar 1 de cada 10 *frames*, xa que presenta un tempo de execución moito máis baixo que para as configuracións anteriores e presenta unha taxa de éxito na detección só un pouco máis baixa que no caso de procesar todos os *frames*. Compre resaltar que non se tiveron en conta opcións nas que se descartasen máis *frames*, xa que isto conlevaría procesar moi poucos frames por cada segundo do vídeo, o cal provocaría que a taxa de éxito nas detección baixase considerablemente.

Tras realizar varias probas do funcionamento do detector coa utilización da fracción de *frames*

a seleccionar que se obtivo, comprobouse que o detector funciona correctamente, xa que ten unha taxa de detección moi alta e detecta os obxectos de interese en prácticamente todas as situacións, devolvendo moi poucos falsos positivos.

6.4.2 Avaliación do intervalo de detección

Realizáronse unha serie de experimentos para comprobar que un obxecto non pode chegar a ser detectado se este non aparece na gravación durante uns 2 segundos, xa que os *frames* que compoñen o intervalo de tempo no que aparece o obxecto en cuestión non conseguirían superar o umbral do PSNR necesario para poder levar a cabo a detección. A partir dos datos obtidos das probas, puídose observar que se o usuario non fixa a súa mirada no obxecto que queira controlar o tempo necesario este non será detectado, polo que o usuario non poderá controlalo.

6.4.3 Omisión de intervalos nos que non hai obxectos para detectar

Cando o usuario fixa a súa mirada nun punto concreto no que non existe ningún obxecto de interese (por exemplo: mentres está a falar ou mirar a alguén), o detector ten que devolver como resultado que non detectou ningún obxecto e non debe volver a executarse ata que o usuario cambie significativamente a rexión que está mirando. Para probar esta casuística, elaboráronse unha serie de vídeos preparados especialmente para cubrir as condicións que se expuxeron e foron procesados polo detector. Tras realizar este experimento, puídose comprobar que o detector non detectou ningún obxecto e evitou entrar na detección ata que o usuario cambiaba o punto ao que estaba mirando.

6.5 Obxectivo do módulo

Tras a correcta execución do módulo de detección de obxectos este devolverá a *label* do obxecto de interese que foi detectado no vídeo de entrada. Estas *labels* serán utilizadas polo nodo central de procesamento de maneira posterior xunto co número de pestanexos para determinar sobre que dispositivo hai que realizar unha acción concreta.

Capítulo 7

Detección de pestanexos

Unha vez se procesou o vídeo de entrada por parte do detector de obxectos e se identificou o dispositivo que o usuario estaba intentando controlar, o seguinte paso é detectar o número de pestanexos que o usuario realiza. Este número de pestanexos é un dato esencial para o correcto funcionamento da aplicación xa que, xunto co obxecto que se detectou, serán os datos que utilizará o dispositivo central para deteminar a acción que se quere levar a cabo sobre un dispositivo determinado. A continuación, vaise a explicar paso a paso o funcionamento deste módulo.

7.1 Datos de entrada

Os datos de entrada que recibe o noso detector de pestanexos consisten nunha serie de arquivos .txt, os cales conteñen a información das EEG do usuario en cuestión. Toda esta información é recopilada por un dispositivo BCI, como xa se comentou no capítulo 3, o cal terá que levar posto o usuario. A finalidade básica deste BCI é recoller a través dos seus electrodos a actividade cerebral da persoa que o leva posto, xa que a partir destes sinais podemos obter os pestanexos que este está a realizar. En concreto, os datos que se almacenan nos ficheiros .txt correspóndense coas mostras do sinal EEG medidas en microvoltios (μV) e tomadas nos instantes de tempo determinados pola frecuencia de muestreo que se estea a utilizar. No caso particular do BCI que se empregou neste proxecto, utiliza unha frecuencia de muestreo de 250 Hz, é dicir, o BCI proporciona 250 mostras do sinal EEG cada segundo. Polo tanto, o número de mostras que se almacenan en cada ficheiro .txt dependerá do intervalo de tempo que se considere, o cal será un parámetro de deseño.
Unha vez se crea o arquivo .txt coas mostras do sinal EEG nun intervalo de tempo, este arquivo será enviado ao noso dispositivo de procesamento central. Deste xeito, o detector de pestanexos irá collendo os ficheiros .txt que lle van chegando para o seu posterior procesamento. Este envío de arquivos será levado a cabo vía MQTT, xa que a comunicación entre os diferentes

dispositivos deste proxecto foi diseñada para realizarse con este protocolo de comunicación. Polo tanto, é preciso dotar tanto ao BCI como ao dispositivo de procesamento central dun chip Wi-Fi, o cal lles permite recibir e enviar calquera tipo de dato mediante MQTT.
Dado que o obxectivo do proxecto é validar a viabilidade práctica do sistema IoT de axuda proposto para persoas con problemas severos de mobilidade, o parámetro de deseño que determina o intervalo de tempo que se considera do sinal EEG en cada arquivo ten que ter un valor relativamente pequeno para emular, na medida do posible, a transmisión de datos en tempo real.

7.2 Procesamento dos datos

Tras rebicir do BCI o arquivo .txt, o nodo central ten que facer unha serie de operacións para procesar o arquivo recibido e, dese xeito, obter a información que nos interesa que, neste caso, sería o número de pestanexos que se reflexan nos datos da entrada. Este procesamento divídese en dúas partes principais, o filtrado e a detección:

7.2.1 Filtrado dos sinais

A información recopilada no arquivo que recibimos como entrada representa as mostras "en cru" do sinal EEG co cal imos ter que traballar pero, para poder traballar máis comodamente con este sinal, primeiro compre aplicar sobre este unha serie de filtros e axustes.
Inicialmente o sinal presenta dous grandes inconvintes para poder ser procesado correctamente. O primeiro consiste en que o sinal presenta unha gran compoñente de continua, tal e como se pode ver na figura 7.1, polo que resulta necesario eliminar esta compoñente de continua para quedarnos coa información que realmente nos interesa.
O outro gran problema é a presenza de ruido ao longo de todo o sinal, como se pode apreciar na figura 7.2. Este ruído ven provocado pola alimentación da placa do BCI. As características hardware do BCI empregado no proxecto fan que este ruído se concentre principalmente na banda de frecuencias entorno aos 50 Hz.
A solución que se implementou no detector para paliar estos problemas consiste en aplicar un filtro paso banda entre 1 e 30 Hz, co que eliminamos a compoñente de continua (0 Hz) e as compoñentes maís ruídosas do sinal entorno a banda de frecuencia dos 50 Hz, que era onde se presentaba o ruido provocado pola alimentación. Decidiuse adoptar esta solución debido a que a actividade cerebral concéntrase no intervalo de 1 a 30 Hz (como se explicou no capítulo 2) polo que, suprimindo a compoñente de continua e os 50 Hz, eliminamos os problemas existentes no sinal e non modificamos en absoluto os datos sobre a actividade cerebral cos que se quere traballar.

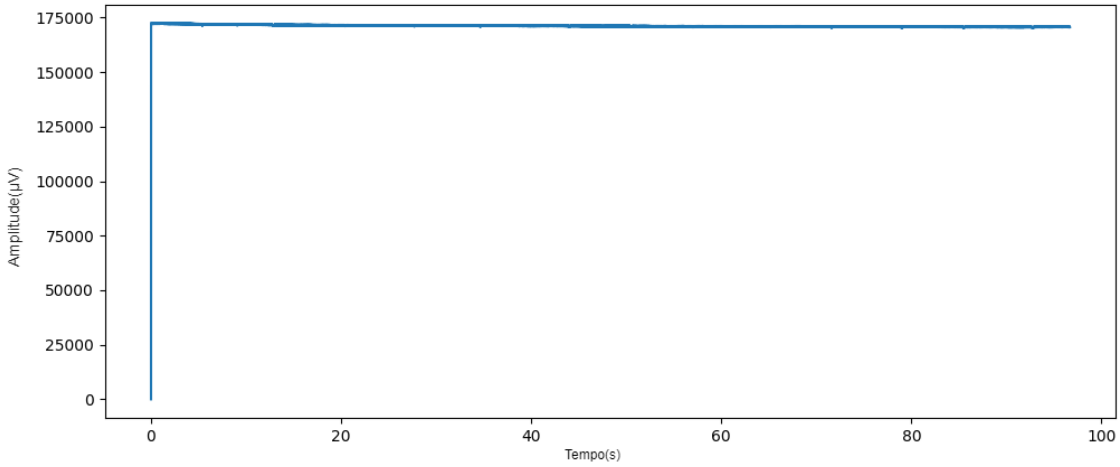

Figura 7.1: Compoñente de continua do sinal.

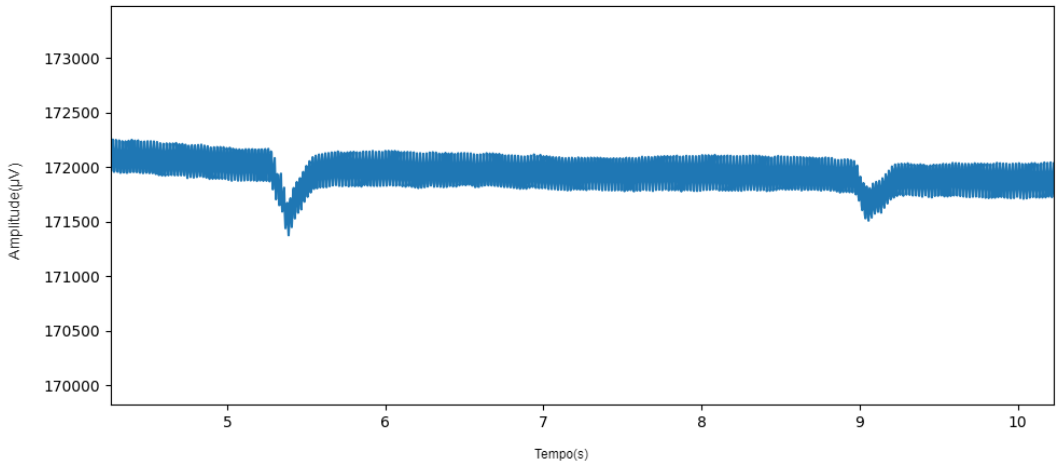

Figura 7.2: Ruído provocado pola alimentación da placa do BCI.

A pesar de aplicar o filtro, o sinal segue presentando un problema. Ao principio dunha gravación, tal e como se mostra na figura 7.3, o sinal presenta moito ruído. Este ruído inicial está provocado tamén pola placa do BCI, a cal realiza sempre unha serie de comprobacións ao principio das gravacións provocando que, nos primeiros segundos, apareza moito ruído no sinal resultante. Polo tanto, e para suprimir este ruído inicial, simplemente decidiuse ignorar os 5 primeiros segundos da gravación. A diferencia dos problemas anteriores, este tipo de artefactos causados polos BCI non supoñen un gran trastorno nun sistema real xa que están moi localizados no tempo (apenas uns segundos) e nunha fase inicial na que o usuario activa o sistema de axuda e non se vai poñer a controlar obxectos de forma inmediata.

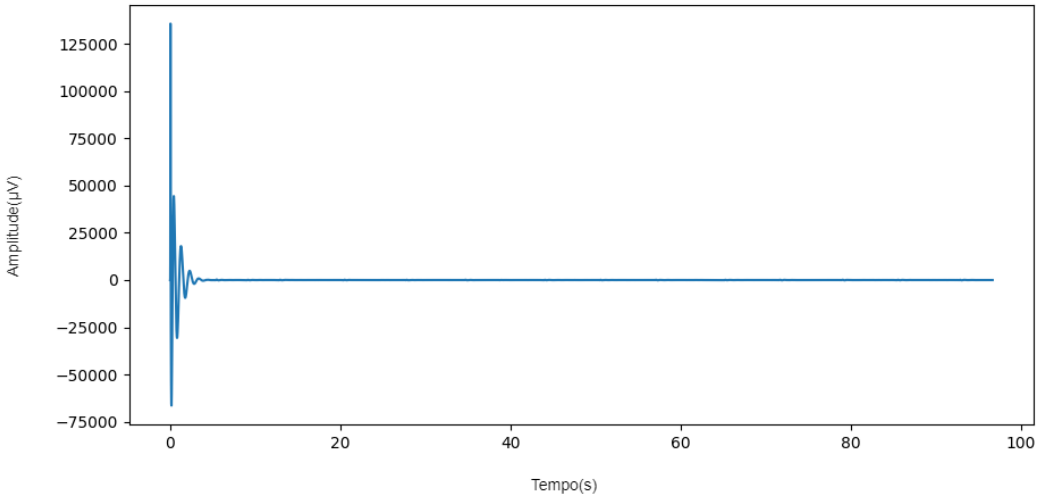

Figura 7.3: Ruido inicial provocado polas comprobacións da placa do BCI.

7.2.2 Detección dos pestanexos

Tras o paso anterior de filtrado do sinal de EEG, o detector vai traballar cun sinal como o que se pode ver na figura 7.4, do cal xa se pode extraer o número de pestanexos. Para levar a cabo esta detección valoráronse e probáronse diversas alternativas:

- Detección no dominio temporal fixando un umbral sobre a amplitude.
- Detección no dominio da frecuencia analizando a enerxía que se concentra na banda correspondente ás ondas Tetha e Delta.
- Detección de cambios abruptos no sinal mediante a transformada Wavelet.

Finalmente, elixiuse levar a cabo a detección no dominio do tempo coa utilización dun umbral debido a que o seu funcionamento era bo, sinxelo e o menos custoso computacionalemente. Polo tanto, para levar a cabo a detección de pestanexos, o detector realiza os seguintes pasos:

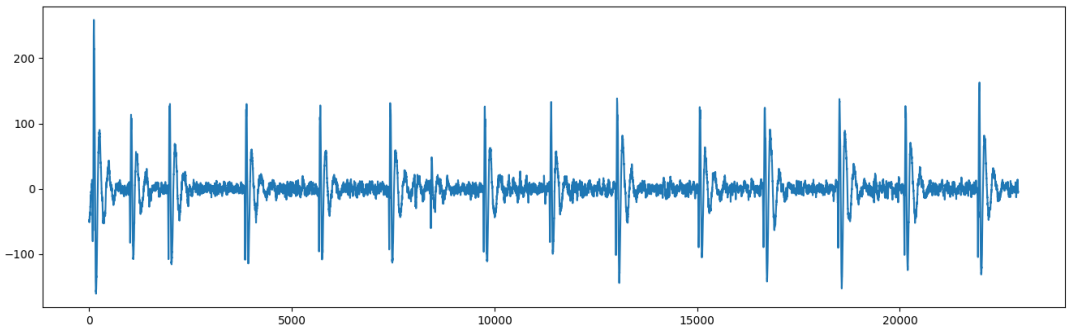

Figura 7.4: Sinal xa filtrado.

1. **Detección baseada en umbrais:** Para detectar un pestanexo no sinal co que se está a traballar, o primeiro paso será analizar a forma na que se representa un pestanexo no propio sinal. Os pestanexos seguen sempre un patrón fixo como o que se ve na figura 7.5. Este patrón consiste en que primeiro se acada un valor negativo na amplitude moi marcado se o comparamos co resto de valores do sinal e, a continuación, un valor positivo da mesma orde. É dicir, un pestanexo voluntario correspóndese realmente cun pico negativo e un pico positivo moi seguidos. Tendo en conta este patrón, decidiuse definir un umbral de forma que, se o sinal supera dito umbral (con valores positivos ou negativos) en dúas ocasións nun intervalo de tempo suficientemente pequeno, asúmese que se produciu un pestanexo. Axustando de forma axeitada o umbral, evítase ademais que os pestanexos involuntarios, os cales alcanzan valores positivos e negativos moito máis pequenos, sexan detectados, evitando así falsos positivos.

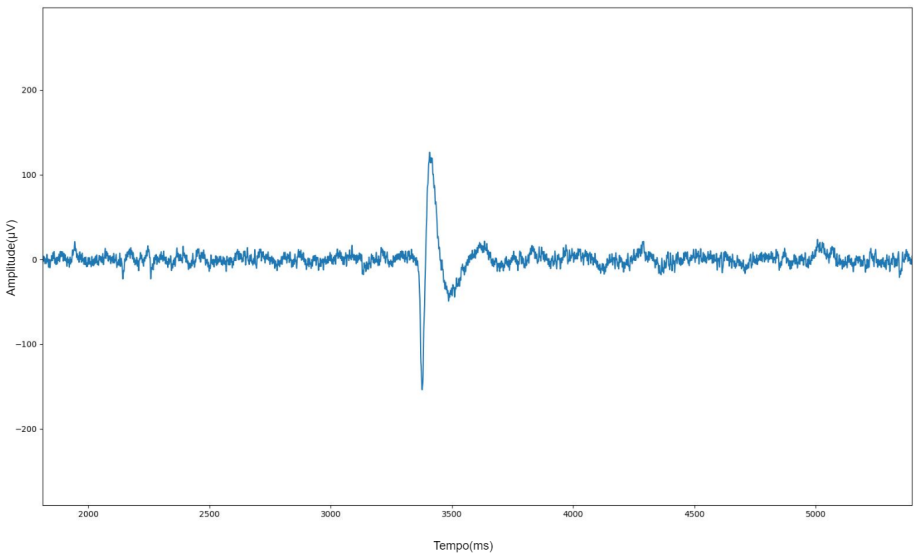

Figura 7.5: Patrón do sinal que representa un pestanexo.

2. **Creación dos checkpoints e límite de tempo:** Tal e como se indicou no primeiro paso, o patrón que estamos detectando consta de 1 pico negativo e 1 pico positivo que sempre se representan seguindo a seguinte orde: **negativo -> positivo**. Polo tanto, para maior robustez do sistema de detección e para evitar que patróns que superen dúas veces o umbral definido sen seguir esa orde fosen detectados como pestenxos, establecéronse uns *checkpoints*, os cales teñen que ser activados na orde establecida para que se conte o pestanexo. Estes *checkpoints* son dúas variables de tipo booleano, as cales están numeradas e non poden ser postas a *true* sen que a anterior o estea tamén, é decir, o *checkpoint* "check2" non pode ter o valor *true* sen que "check1" teña tamén un valor true. Polo tanto, o pestanexo tense en conta no momento en que o segundo *checkpoint* ponse a *true*.

   Por outra parte, necesítase establecer un límite de tempo para detectar un pestanexo. Neste caso, estableceuse unha variable *intervalo_pest*, a cal determina o límite de tempo para detectar un pestanexo a partir de que se supere o primeiro *checkpoint*. En caso de que ese tempo remate antes de superar o derradeiro *checkpoint*, todas as variables correspondentes aos *checkpoints* volverán poñerse a *false* para que outros pestanexos poidan ser detectados ao longo do sinal.

3. **Reconto dos pestanexos:** Como se explicou nos capítulos anteriores, poden existir certos obxectos sobre os que se poden aplicar diferentes tipos de accións considerando distintas secuencias de pestanexos. Por esa razón, resulta importante contabilizar o número de pestanexos que se producen tras a detección dun obxecto. Para isto, o usuario contará cun intervalo de tempo prefixado polo sistema para realizar dita secuencia. En caso de que os pestanexos se produzan fora deste intervalo o detector non os terá en conta. Para contabilizar os pestanexos que se detectan a partir do umbral e os *checkpoints*, creouse un contador *num_pest* que se inicializa a cero no momento no que se detecta un obxecto e se entra no módulo de detección de pestanexos. Este contador vaise incrementando cada vez que se detecta un novo pestanexo. Desta forma, esta variable conterá o número de pestanexos totais ao finalizar o intervalo de tempo no que se está procesando o sinal EEG.

7.3 Decisións de deseño

Nesta sección vanse a aclarar e xustificar as decisións de deseño que se levaron a cabo durante o desenvolvemento deste módulo.

7.3.1 Definición do umbral para a detección de pestanexos

Para detectar os pestenxos que se producen no sinal vaise facer uso dun umbral, o cal adoptará un valor positivo e negativo para comprobar se o que se está a detectar no sinal é un pestanexo ou non. Para definir o valor deste umbral levouse a cabo un análise estadístico, o cal se pode dividir en dúas partes:

1. **Obtención do histograma:** O primeiro paso consistíu en xerar un histograma a partir dos datos correspondentes a un banco de sinais de EEG do que xa se dispoñía. Este banco de snais contiña mostras de diferentes individuos obtidas nunha serie de experimentos nos que se lle pedía que alternaran períodos onde tiñan que pestanexar con períodos onde permanecían inmóviles simplemente cos ollos abertos. O histograma resulta principalmente útil para analizar a distribución dos valores da amplitude do sinal, é decir, permítenos coñecer os valores máis habituais do sinal e visualizar os valores atípicos, aqueles cunha amplitude significativamente maior (ou menor) ao resto do sinal. A partir de este tipo de información, xa se podería definir "a ollo" un umbral bastante axustado para detectar os pestanexos e minimizar falsas detecciónss.

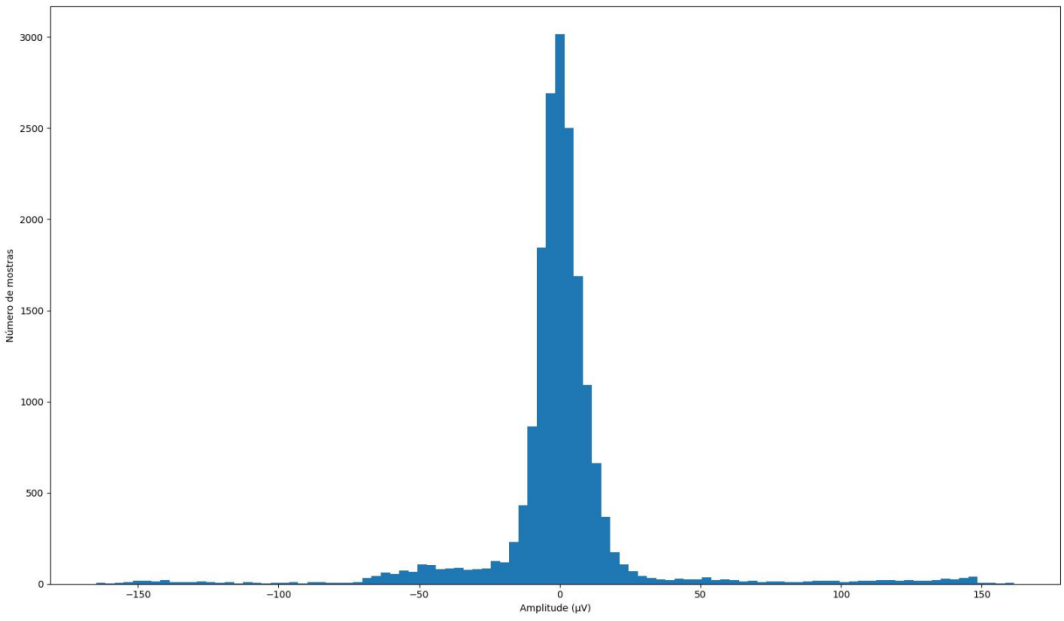

Figura 7.6: Histograma.

2. **Obtención do umbral:** Unha vez temos o histograma podemos ver a distribución de valores que se dá nos sinais. Sen embargo, resulta máis sistemático aplicar un método estadístico sobre este conxunto de mostras para definir o valor do umbral. Dado que os pestanexos xeran amplitudes moi elevadas no EEG con respecto aos seus valores

medios, os valores atípicos que se producen no sinal están directamente relacionados co feito de que a persoa realizou un pestanexo voluntario. Deste xeito, estos valores poden ser considerados como *outliers* e podemos aplicar algunhas das técnicas estadísticas máis comúns para detectar este tipo de valores.

En concreto, decidiuse empregar a técnica baseada no chamado InterQuartil Range (IRQ), que consiste basicamente en determinar os diferentes cuartiles para a distribución de datos que queremos analizar e fixar os umbrais a partir dos cales se consideran os valores atípicos como:

$$u_1 = Q_1 - 1,5 * \text{irq}$$
$$u_2 = Q_3 + 1,5 * \text{irq},$$

onde Q_1 e Q_3 representan o primeiro e o terceiro cuartil, respectivamente, e irq correspóndese co rango intercuartil e calcúlase como $\text{irq} = Q_3 - Q_1$.

Aplicando este método, obtiveronse valores similares para ambos umbrais (positivo e negativo), como era de esperar, polo que decidiu fixar un único umbral para a detección de pestanexos, que neste caso foi de 60.

Para avaliar a adecuación do valor fixado para este umbral, vanse facer unha serie de probas nas que se utilizarán as métricas *Recall* e Precisión, que serven para determinar a cantidade de pestanexos presentes nas sinais de maneira exitosa e a taxa de detección de pestanexos que foron detectados e non deberían de ter sido detectados. Con estas dúa métricas pódese analizar como de ben funciona o detector de pestanexos en función dos valores que teña o umbral.
Os motivos polos que este método de análise é eficaz para determinar o umbral son que temos unha gran cantidade de datos para poder realizar probas e que os sinais non varían moito entre diferentes persoas, polo que estes valores son válidos para calquera usuario. No caso de que se queira afinar máis o valor deste umbral para un usuario concreto, este mesmo procedemento podería levarse a cabo mediante a realización dun período de *training*, no cal se recopilarían suficientes mostras específicas do sinal do usuario para determinar de forma particular o umbral.

7.3.2 Intervalo de tempo para detectar un pestenexo

A partir das diferentes mostras cas que se traballou puídose observar que un pestenexo voluntario é representado nos sinais durante un intervalo de tempo que adoita rondar os 2 segundos, polo que se decidiu que, neste proxecto, o intervalo de tempo que se establece para a detección dun pestanexo sexa desta mesma duración.

7.3.3 Intervalo de tempo para detectar secuencias pestenexos

Neste proxecto estableceuse un intervalo máximo de tempo de 10 segundos para detectar unha secuencia determinada de pestanexos, coa cal se indicará a acción que se quere executar no dispositivo a controlar. Hai que ter en conta que esta xanela de tempo comeza no momento no que se detecta o primeiro pestaxo. As secuencias utilizadas na aplicación poden estar formadas por entre 1 e 4 pestanexos. Polo tanto, é importante definir un intervalo de tempo suficiente para poder detectar toda a secuencia, pero sen que este valor sexa demasiado grande xa que podería levar a tempos de espera no sistema non desexados.
Tendo en conta esto, este intervalo de tempo foi determinado a partir do tempo establecido para realizar un pestanexo e do número máximo de pestanexos que poden conter as secuencias. No caso particular dunha secuencia que conta con 4 pestanexos, o tempo mínimo necesario para detectar toda a secuencia será duns 6 segundos (3 pestanexos despois do primeiro). Sen embargo, hai que ter en conta que unha persoa pode tardar máis de 8 segundos en realizar unha secuencia completa de 4 pestanexos. Polo tanto, decidiuse fixar a xanela temporal para a detección en 10 segundos para ter en conta tanto o tempo mínimo que tardará en realizarse a secuencia de pestanexos máis grande, como outros factores como o tempo de reacción ou a rapidez do usuario.

7.4 Probas do módulo

Nesta sección vanse a expoñer as diferentes probas que se levaron a cabo para probar o correcto funcionamento do módulo, comprobando que se cumpran os diferentes requisitos que foron definidos ao inicio do proxecto.

7.4.1 Definición do umbral

Para avaliar o funcionamento do sistema de detección en función do valor que se estableza para o umbral, decidiuse levar a cabo un experimento no que se determinaran os valores que se obterían para as métricas de Precisión e *Recall*, considerando diferentes umbrais. Para realizar este experimento, utilizouse o banco de sinais de EGG do que se dispoñía e determinouse a tasa de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos. A partir dos datos obtidos, determináronse os valores da métricas anteriores.

Na táboa 7.1, móstranse os resultados obtidos para a métrica de Precisión. Estes datos calcularonse tendo en conta que esta métrica se define como precisión = TP/(TP + FP), onde TP son os verdadeiros positivos e FP os falsos positivos. Polo tanto, esta métrica proporciona a porcentaxe de detecciónss exitosas.

Tal e como se pode apreciar na táboa 7.1, a métrica Precisión alcanza o valor máximo de 1

Valor do umbral	30	45	60	90	120	150
Precisión	0,35	0,89	0,97	1	1	1

Táboa 7.1: Datos da métrica de Precisión.

cando se emprega un umbral igual ou superior a 90, se ben xa se obtén un valor moi próximo a 1 cun umbral entorno a 60. Ademais, resulta evidente que valores de umbral inferiores a 60 no son adecuados xa que a detección dos pestanexos comeza a fallar.

Na táboa 7.2, preséntanse os valores obtidos para o caso da métrica *Recall*, que se calculará do mesmo xeito que se definiu no capítulo anterior.

Valor do umbral	90	120	150
Recall	1	0,98	0,92

Táboa 7.2: Datos da métrica *Recall*.

Como se pode apreciar, esta métrica proporciona valores altos próximos a 1 para a maioría de valores que se probaron para o umbral. Tan só cando se escolle un valor moi alto e próximo ao valor de pico que se alcanza nos pestanexos, esta métrica comeza a baixar. Tendo en conta os resultados mostrados nas dúas tablas anteriores, podemos concluír que o valor de 60 proporcionado polo método IRQ para definir o umbral pode ser demasiado conservador e, que realmente, sería máis aconsellable empregar un valor máis próximo a 90. A definición do valor do umbral de forma empírica neste caso seguramente sexa máis adecuado, polo que sería conveniente explorar outros métodos estadísticos que nos proporcionen valores máis axustados.

Sen embargo, outra reflexión interesante que se pode facer a partir dos resultados obtidos é que o sistema de detección non é excesivamente sensible ao valor do umbral. De feito, calquera valor entre 60 e 120 produciría detección bastante precisas.

7.5 Procesamento de diferentes EEG

A seguinte proba que se realizou consistiu en comprobar se o detector funcionaba adecuadamente con diferentes secuencias de pestanexos para o umbral establecido (fixado a 90 por seguridade) e coa xanela temporal de 10 segundos para a detección. Para iso, procesáronse diferentes gravacións de EEG que contiñan diferentes secuencias de pestanexos, é dicir, diferente número de pestanexos seguidos, e comprobouse o número de pestanexos que devolvía o detector. Tras varias execucións púidose comprobar que o detector funcionaba correctamente, xa que devolveu unha taxa de acerto do 100%. Finalmente, compre destacar que esta

taxa pode variar cando o detector sexa usado nun entorno real, xa que a taxa actual obtívose a partir de gravacións creadas nun entorno de probas controlado.

7.6 Obxetivo do módulo

O obxetivo principal do módulo é detectar o número de pestanexos que realizou o usuario mediante o procesamento dos sinais EEG enviadas polo BCI. O número de pestanexos obtidos será enviado ao dispositivo de procesamento central para que este poida determinar a acción que se ten que levar a cabo no dispositivo que se queira controlar, detectado previamente polo detector de obxectos.

Capítulo 8

Conectividade

Tal e como se víu nos capítulos anteriores, para que a aplicación funcione correctamente necesitamos crear unha entorna no que varios dispositivos se comuniquen entre si. Para levar isto a cabo necesitamos definir e implementar un módulo de conectividade que permita a este conxunto de dispositivos comunicarse, tendo en conta que presentan grandes deferencias entre si tanto a nivel de hardware como a nivel de software. Neste capítulo expoñerase a solución que se escolleu para implementar o módulo de conectividade e explicaranse as diferentes partes que o conforman.

8.1 O protocolo MQTT

A día de hoxe existen multitude de tecnoloxías e protocolos para crear unha comunicación ou realizar intercambios de arquivos entre diferentes dispositivos. Algúns como Bluetooth ou NFC levan presentes moito tempo nas nosas vidas e son moi utilizados pero, para este proxecto, vaise a utilizar un protocolo de comunicación que está a gañar moita popularidade nestes últimos anos. O protocolo elixido foi MQTT, un protocolo de comunicación M2M basado na pila TCP/IP que está gañando unha gran popularidade debido a que é moi utilizado polos dispositivos que forman parte de sistemas IoT.

8.1.1 ¿Por que MQTT?

O motivo principal polo que se escoulleu o protocolo MQTT frente a outros protocolos e tecnoloxías é o conxunto de vantaxes que este protocolo presenta. Por un lado, temos todas as vantaxes que presenta o uso do patrón publicador/subscritor, como a escalabilidade e o desacoplamento dos clientes e, por outro, MQTT presenta unha serie de características que o fan sobresaír sobre os seus competidores. Entre estas características destaca a súa sinxeleza e lixeireza, o que fai que sexa un protocolo perfecto para dispositivos de escasa potencia ou

que dependen dunha batería para a súa alimentación, xa que utiliza moi poucos recursos e, polo tanto, isto tradúcese nun menor consumo de enerxía. Tamén compre destacar que, debido a súa lixeireza, require un ancho de banda mínimo, unha propiedade moi apreciada nas redes sen fíos ou en conexións con posibles problemas de calidade. Finalmente, MQTT dispón dunha gran variedade de funcionalidades adicionais, como mecanismos de seguridade e a posibilidade de definir diferentes niveis na calidade do servicio (QoS). Por último, trátase dunha solución sobradamente testada que aporta robustez e fiabilidade á conectividade dos dispositivos involucrados no proxecto.

8.1.2 Funcionamento de MQTT

O funcionamento de MQTT basease no patrón publicador/subscritor (pub-sub), no cal interveñen 3 tipos de dispositivos diferentes: publicadores, subscritores e *brokers/servers*.

8.1.3 Publicadores

Os publicadores son un tipo de cliente que se conecta a un *broker* dunha rede MQTT co obxetivo principal de publicar unha serie de mensaxes ou arquivos baixo un *topic* determinado. No noso proxecto, o papel de publicador será levado a cabo polo BCI e pola cámara da vídeo, xa que o seu cometido principal é a recopilación de información (vídeos no caso da cámara e sinais EEG no do BCI) para o seu posterior envío ao nodo de procesamento central. Máis adiante, tamén desempeñará este papel o nodo central, xa que, unha vez determine a acción que se quere levar a cabo sobre o dispositivo detectado, terá que enviar unha mensaxe ca orde correspondente ao dispositivo destino.

8.1.4 Subscritores

Os subscritores son outro tipo de cliente que se conecta a un *broker* dunha rede MQTT, pero o obxectivo da súa conexión é crear unha subscrición a un *topic*. Con esta subscrición, o *broker* enviaralle ao subscritor todas aquelas mensaxes ou arquivos que cheguen ao *broker* baixo o *topic* ao que se subscribíu. No noso proxecto, o papel de subscritor será desempeñado polo nodo central para a recepción dos arquivos que sexan publicados por parte da camára e do BCI, e tamén polos dispositivos que se queiran controlar coa nosa aplicación, xa que necesitaremos que recollan a mensaxe publicada por parte do nodo central ca orde que teñen que levar a cabo.

8.1.5 Brokers/Servers

Os *brokers* ou *servers* son os dispositivos dunha rede MQTT encargados de procesar as mensaxes enviadas polos publicadores e enviar ditas mensaxes aos subscritores que estean subscritos

ao *topic* definido para as mensaxes. Hai dous tipos de *brokers* MQTT, online e locais. Os brokers online son aqueles que están dispoñibles para o seu uso a través de internet, mentres que os locais son un tipo de broker creado localmente para o seu uso por parte dos dispositivos que pertenzan á rede local que se quere formar. Tendo en conta esto, no noso proxecto vamos a utilizar un *broker* offline, o cal será executado no portátil que desempeñará tamén o rol de nodo central de procesamento.

8.2 Requisitos hardware/software

Tal e como se comentou neste capítulo, o protocolo MQTT está baseado na pila TCP/IP, polo que para a súa utilización necesitase facer uso dunha tecnoloxía que permita unha transmisión física de datos. No caso deste proxecto vaise facer uso da tecnoloxía Wi-Fi, polo tanto, para que todos os dispositivos do entorno do noso proxecto poidan facer uso de MQTT, estes teñen que contar cun chip Wi-Fi. En caso de que estes dispositivos non tivesen integrado este chip no seu hardware, poderíase facer uso dun adaptador Wi-Fi se contan cun porto USB.

Con respecto aos requisitos software, simplemente se terá que instalar unha implementación de MQTT nos dispositivos que vaian a utilizar o protocolo. Neste proxecto optouse por utilizar a implementación OpenSource Eclipse Mosquitto [34], a cal se pode instalar en diferentes sistemas operativos mediante un instalador ou por liña de comandos. Ademais, se se quere crear un *script* para executar os subscritores e publicadores con Python, necesítase instalar a librería paho-mqtt. No caso de utilizar a liña de comandos, estes serían os comandos necesarios para a instalación de todo o software previamente mencionado:

1. sudo apt-get install mosquitto
2. sudo apt-get install mosquitto-clients
3. pip3 install paho-mqtt

8.3 Implementación no proxecto

Para habilitar a utilización do protocolo MQTT nos diferentes dispositivos e elementos do noso proxecto, creáronse dous *scripts* en Python usando a librería paho-mqtt. O programa sub.py define o comportamento das compoñentes que actuarán como subscritores e, polo tanto, será executado no nodo de procesamento central e nos dispositivos que se queiran controlar, mentres que o programa pub.py define o comportamento dos publicadores polo que será executado no nodo de procesamento central, na placa incorporada na cámara e no BCI.

8.3.1 Programa pub.py

O programa pub.py ten como cometido principal implementar o funcionamento dos publicadores existentes no noso proxecto. Este pequeno programa seguirá sempre os seguintes pasos:

- **Creación do cliente:** Créase o publicador que vai conectarse á rede MQTT.
- **Conexión co *broker*:** Unha vez o publicador xa está creado, este ten que conectarse ao *broker* da rede MQTT, especificando a súa IP ou url (no caso dun servidor online) e o porto ao que queira conectarse que, por defecto, adoita ser o 1883 para comunicación non cifrada e o 8883 para comunicación cifrada.
- **Preparación do *payload*:** Tras establecer a conexión hai que preparar o *payload* da mensaxe que se enviará ao *broker*. Se a mensaxe é unha cadea de texto simple, esta non necesitará ningún tipo de preparación pero, no caso de querer enviar un arquivo como *payload* da mensaxe, terase que obter o seu contido e pasarse a un formato de *byteArray*, xa que é un dos tipos de datos cos que MQTT adoita traballar.
- **Publicación da mensaxe:** Tras preparar o contido do *payload* da mensaxe, procederase á publicación da mesma, indicando o *topic* no que se quere publicar a mensaxe e o nivel de QoS que se desexa utilizar.
- **Desconexión:** Unha vez se publique a mensaxe, o publicador pechará a conexión, a cal poderá retomar cando desexe facer outra publicación.

8.3.2 Programa sub.py

O programa sub.py ten como cometido principal implementar o funcinamento dos subscritores existentes no noso proxecto. Este pequeno programa seguirá sempre os seguintes pasos:

- **Creación do cliente:** Crease o subscritor que vai conectarse á rede MQTT.
- **Conexión co *broker*:** Unha vez o subscritor xa está creado, este ten que conectarse ao *broker* da rede MQTT, especificando a IP (ou url) do *broker* e o porto ao que queira conectarse que, como no caso anterior, adoita ser o 1883 para comunicación non cifrada e o 8883 para comunicación cifrada.
- **Creación da subscrición:** Unha vez se estableceu a conexión, o subscritor procederá a crear unha subscrición, establecendo o *topic* ao que quere subscribirse e o nivel de QoS que se desexa utilizar.

- **Recepción das mensaxes:** Tras realizar todos os pasos anteriormente mencionados, o subscritor recibirá todas aquelas mensaxes que sexan publicadas no *topic* ao que se subscribíu. No caso de recibir unha cadea de texto, tan só terá que recoller dita cadea pero, en caso de recibir o contido dun arquivo, o subscritor terá que crear un novo arquivo (ao que se lle asignará un nome) nun path do dispositivo no que se está a executar e volcar o contido do *payload* da mensaxe en dito arquivo. Desta forma, crearase unha copia do arquivo que se publicou.
- **Desconexión:** Unha vez se estableza unha conexión co *broker*, o subscritor manterá dita conexión á espera de recibir mensaxes. Só se producirá unha desconexión en caso de que se realice de maneira manual, se se apaga o dispositivo que está a executar o subscritor ou se se produce un erro na conexión.

8.4 Probas do módulo

Nesta sección vanse a expoñer as diferentes probas que se levaron a cabo para avaliar o correcto funcionamento do módulo, comprobando que se cumpran os diferentes requisitos que foron definidos ao inicio do proxecto.

Para probar o funcionamenteo deste módulo, simplemente se procedeu a lanzar un *broker*, dous clientes subscritos a diferentes *topics* e dous publicadores MQTT. Unha vez estes dispositivos estaban operativos, realizáronse unha serie de publicacións a través dos publicadores e comprobouse que os subscritores recibisen correctamente as mensaxes que tivesen asociado un *topic* ao que estivesen subscritos. Ademais, consideráronse diferentes tipos de mensaxes, incluíndo contido multimedia, arquivos de texto e arquivos de datos estruturados. Tras realizar 20 publicacións, 10 con cada publicador e 10 de cada *topic*, comprobouse que a os subscritores recibiron con éxito todas as mensaxes, verificando deste xeito que o módulo de conectividade funcionaba correctamente.

Capítulo 9

Dispositivo central de procesamento

Unha vez xa solucionamos o problema da conectividade e creamos os detectores adecuados para obter o número de pestanexos e recoñecer os obxectos que se queiran controlar, o seguinte paso é a implementación dun programa que faga uso dos detectores para obter, a partir dos datos enviados pola cámara e o BCI, o comando que se ten que enviar ao dispositivo que se queira controlar. Este programa simula, por tanto, o conxunto de accións que tería que levar a cabo un nodo central de procesamento para integrar a información que recibe dos dispositivos encargados de adquirir os datos (BCI e cámara) e decidir a orde que se vai enviar ao dispositivo sobre o que se quere actuar.
Neste capítulo vaise a explicar o funcionamento deste programa, o cal pode ser executado, por exemplo, nun portátil que simula o rol de dispositivo central.

9.1 Entrada do programa

O primeiro paso consiste en cargar o arquivo .txt que enviará o BCI cos datos da EEG e a trama de datos correspondente ao vídeo que se irá enviando desde a cámara. Estes arquivos serán recibidos polo dispositivo de procesamento central mediante a execución do código de subscritor MQTT e serán almacenados nun path concreto do directorio de arquivos polo que, para obtelos, simplemente se precisa indicarlle ao programa ese path.

9.2 Detección dos obxectos

Unha vez se obteñen os arquivos de entrada, os *frames* do vídeo teñen que pasarse ao detector de obxectos para o seu procesamento. O código do detector xa foi integrado nunha función que recibe como entrada o vídeo a procesar e devolve o obxecto de interese detectado ou *null*. Desta forma, o nodo central de procesamento simplemente ten que facer unha chamada á

función de detección con cada trama de datos que vai recibindo.
No caso de detectar un obxecto, hai que comprobar se o este é un obxecto de combinación ou non, xa que nalgúns casos para controlar un dispositivo concreto téñense que detectar dous obxectos de forma consecutiva. Para iso, comprobarase se dito obxecto está presente na lista de obxectos de combinación, a cal estará definida de antemán. En función do resultado desta comprobación, pódense dar tres situacións:

- **O obxecto non está presente na lista:** Se o obxecto detectado non pertence á lista de obxectos de combinación non será necesario detectar un segundo obxecto, polo que o seguinte paso será a detección de pestanexos.
- **O obxecto está presente na lista:** Se o obxecto está presente na lista terase que seguir co procesamento do vídeo para ver se se detecta outro obxecto que sexa compatible co detectado previamente. Para levar isto a cabo, volverase chamar á función do detector de obxectos pero, agora, haberá que pasarlle como parámetros os seguintes *frames* do vídeo que se reciben, o obxecto que se detectou previamente (para evitar que se detecte o mesmo outra vez) e o número de *frames* que se quere procesar, debido a que se lle vai a dar un tempo límite para detectar o obxecto de combinación. Este número de *frames* ven determinado directamente polo parámetro fps do vídeo e polo límite de tempo establecido que será un parámetro do sistema.

  O obxecto detectado nesta segunda fase de detección terá que ser compatible co primeiro obxecto detectado para conformar unha combinación válida na lista de obxectos de combinación. Se a combinación é válida pasarase a detectar os pestanexos pero, en caso contrario ou se non se detectou ningún obxecto, a execución do programa volvera ao paso inicial.
- **Non se detectou ningún obxecto:** Se no vídeo de entrada non se detecta ningún obxecto, a execución do programa volverá ao paso inicial.

9.3 Detección de pestanexos

Cando se detecta un obxecto válido, é dicir, que está na lista de obxectos que se poden controlar ou na lista de obxectos de combinación, comeza a procesarse a información correspondente ao EEG da persoa que está utilizando o sistema. Esta información é recibida polo nodo central a través dos arquivos .txt que lle chegan con MQTT. De xeito similar ao que acontecía co detector de obxectos, todo o código necesario para a detección de pestanexos foi integrada nunha función que recibe como parámetro o arquivo .txt que se quere procesar e que devolve o número de pestanexos detectados no intervalo de tempo prefixado para a detección. De novo, poden ocorrer dúas situacións:

- **Non se detectan pestanexos** voluntarios ou o **número de pestanexos non coincide con ningunha secuencia definida** para o obxecto detectado. Neste caso, asúmese que o usuario non quere realizar ningunha acción específica sobre o obxecto e vólvese ao punto inicial.

- **Detéctase un número de pestanexos correcto**. Neste caso, o nodo central pasaría ao seguinte paso que sería determinar que acción se quere realizar sobre o obxecto detectado.

9.4 Selección da acción a realizar

Agora que xa se dispón do obxecto/s detectado/s e do número de pestanexos, xa podemos determinar a partir destas variables que dispositivo se quere controlar e que acción se quere levar a cabo sobre él.

Na táboa 9.1, pódese ver unha lista con algúns dos obxectos que se poden controlar (incluíndo de combinación) coas correpondentes accións que se poden executar sobre eles. En particular, na terceira columna da táboa índicase o obxecto que a persoa ten que mirar e na primeira a secuencia de pestanexos necesaria para realizar unha determina acción relacionada co obxecto detectado. O obxecto a controlar fai referencia ao dispositivo que ten que executar a acción. Por exemplo, no primeiro caso, se a persoa mira un tenedor e realiza dous pestanexos, automaticamente o sistema enviaría a través do móbil unha mensaxe ao seu coidador indicando que ten fame.

Número de pestanexos	Obxecto a controlar	obxecto detectado	Acción a executar
2	Móbil	Tenedor	Enviar mensaxe de que o usuario ten fame
2	Móbil	Copa	Enviar mensaxe de que o usuario ten sede
2	Móbil	Silla	Enviar mensaxe de que o usuario quere sentarse
3	Móbil	Silla	Enviar mensaxe de que o usuario quere levantarse da cama
2	Móbil	Móbil	Chamar ao contacto de emerxencia
2	TV	TV + mando	Pasar canal
3	TV	TV + mando	Encender TV
4	TV	TV + mando	Apagar TV

Táboa 9.1: obxectos a controlar + accións.

Coa información proporcionada pola función de detección de obxectos e pola función de detección de pestanexos, o programa correspondente ao nodo central simplemente ten que consultar a lista anterior e determinar a acción que se vai a realizar.

9.5 Envío da orde

Tras determinar o dispositivo que se quere controlar e a orde que se lle ten que enviar, o derradeiro paso no dispositivo central de procesamento é enviar dita orde en forma de comando ao dispositivo final. Para levar a cabo esta acción, o dispositivo central fará uso do código de publicador MQTT, co que enviará dito comando no *payload* da mensaxe. Ao mesmo tempo, o obxecto que se queira controlar terá que estar executado o código de subscritor MQTT para así recibir aquelas mensaxes que sexan publicadas polo dispositivo central cun *topic* especificamente creado para el. Cando o dispositivo destino reciba a mensaxe, executará o comando enviado no *payload*, cumplindo así coa orde que o usuario quería que este levase a cabo.

9.6 Fluxo de execución

Nesta sección vaise a ilustrar o fluxo de accións que ten que levar a cabo usuario para controlar un dispositivo concreto:

1. O usuario fixa a súa mirada durante uns segundos sobre o dispositivo que queira controlar, por exemplo un móbil.
2. A información do vídeo chega ao nodo central de procesamento, o cal determina entrar na fase de detección de obxectos e detecta o móbil no vídeo.
3. Unha vez o usuario xa fixou a súa mirada sobre o móbil este realizará unha secuencia de pestanexos de, por exemplo 2 pestanexos, os cales serán gravados no EEG que se enviará ao nodo de procesamento central nun arquivo .txt.
4. A información do EEG chega ao nodo de procesamento central nun arquivo .txt, o cal será procesado para determinar a secuencia de pestanexos que realizou o usuario.
5. Tras obter o obxecto detectado e a secuencia de pestanexos realizada, estes datos serán utilizados para determinar que, neste caso particular, se quere controlar o móbil e realizar a acción de chamar a un contacto de emerxencia.
6. Finalmente o nodo de procesamento central encargarase de determinar o comando correcto para levar a cabo a acción, a cal será enviada ao móbil para que a execute.

9.7 Probas unitarias e de aceptación

Tras a finalización deste módulo a aplicación estará completamente finalizada, polo que o último paso que queda consiste na realización de probas tanto unitarias como de aceptación.

9.7.1 Probas unitarias

As probas unitarias consisten nunha serie de probas independentes entre si que nos permiten comprobar que o funcinamento dos módulos implementados é o adecuado. Polo tanto, para probar a aplicación, realizáronse unha serie de probas para cada módulo existente, comprobando que estes obteñan un resultado determinado en función dos parámetros de entrada que se lles proporcione, xa que ditos parámetros de entrada determinarán a casuística que se queira probar. Ademais, implementáronse unha serie de probas para comprobar que a integración dos diferentes módulos é correcta.

A aplicación conseguíu pasar exitosamente as probas establecidas, polo que se pode afirmar que os módulos que compoñen o proxecto funcionan correctamente tanto de maneira individual como de forma conxunta.

9.7.2 Probas de aceptación

As probas de aceptación consisten en que o cliente que encargou a aplicación probe a versión final da mesma. Como este proxecto se levou a cabo como un traballo de fin de grao, non se conta explicitamente cun cliente, polo que a aplicación foi probada polos directores deste proxecto: Óscar Fresnedo Arias e Francisco Laport López.

Ambos directores probaron a aplicación e comprobaron que se cumplisen todos os requisitos funcionais e non funiconais, os cales foron definidos ao principio do proxecto. Tras a aprobación pola súa parte considerouse que estas probas foron superadas.

Capítulo 10

Conclusións e traballo futuro

Neste capítulo explicaranse as conclusións as que se chegou tras a realización do proxecto, así como posibles melloras que se poderían levar a cabo sobre o proxecto actual.

10.1 Conclusións

O proxecto realizado tiña como obxetivo principal desenvolver unha aproximación software dun sistema de IoT de axuda a persoas con mobilidade reducida para poder avaliar a viabilidade técnica do mesmo. Tendo este obxectivo como premisa inicial levouse a cabo o desenvolvemento dunha aplicación capaz de permitir a un usuario controlar diferentes dispositivos cotiás da súa contorna simplemente facendo uso da súa mirada e dos seus pestanexos. En consecuencia, conseguíuse realizar unha aproximación para unha ferramenta de gran valor social, debido á súa importancia e utilidade para persoas con dificultades motrices, otorgándolles un certo grao de autonomía.

A través da realización deste proxecto tívose que levar a cabo unha apredizaxe sobre conceptos asociados á detección de obxectos mediante algoritmos de *machine learning*, ao análise da actividade cerebral dunha persoa mediante a utilización de interfaces cerebro-ordenador ou á utilización de protocolos de comunicación para o envío e recepción de arquivos de maneira inalámbrica. Esta aprendizaxe resultoume francamente útil, xa que me permitíu descubrir e comprender tecnoloxías e campos da informática que non tivera a oportunidade de estudar ao longo da carreira. Ademáis, reforzáronse e ampliáronse coñecementos sobre análise, deseño e desenvolvemento de proxectos que foron adquiridos durante o grao a través de varias asignaturas.

10.2 Traballos futuros

Debido a que este proxecto foi considerado como unha primeira aproximación software do produto, unha posible liña de traballo futuro sería a implementación dun prototipo hardware con estas mesmas funcionalidades e baseado nos módulos empregados neste proxecto. Esto permitiría implementar a nosa ferramenta software en dispositivos hardware compatibles cos que facer que todo mundo puidese beneficiarse das funcionalidades do produto software que se desenvolveu.

Por outra banda, este proxecto foi implementado de maneira que sexa sinxelo realizar calquera tipo de modificación, polo que tamén se poderían implementar melloras e novas probas, as cales serían moi benificiosas de cara a mellorar a viabilidade do software, xa que todas as probas que se levaron a cabo ata o momento foron realizadas nun espacio de probas controlado.

Apéndices

Relación de Acrónimos

AMQP *Advanced Message Queuing Protocol.* 15

BCI *Brain-Computer Interface.* 1, 13, 20, 21, 36

ECG Electrocardiograma. 19

EEG Electroencefalografía. 3, 9, 13, 19

EMG Electromiograma. 19

FPS *frames per second.* 44

IoT *Internet of Things.* 15

M2M *Machine to Machine.* 15

MQTT *MQ Telemetry Transport.* 15, 36

PSNR *Peak Signal-to-Noise Ratio.* ii, 8, 40, 45–47, 49

R-CNN *Regions with Convolutional Neural Networks.* 5, 18

SSD *Single Shot Multibox Detector.* 5, 18

WAMP *Web Application Messaging Protocol.* 15

YOLO *You Only Look Once.* 5, 18, 45

Bibliografía

[1] Funcionamento do algortimo yolo. [En liña]. Dispoñible en: https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

[2] Informacion das redes neuronais. [En liña]. Dispoñible en: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

[3] Información sobre mqtt. [En liña]. Dispoñible en: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[4] Información sobre bci´s. [En liña]. Dispoñible en: https://www.emotiv.com/bci-guide/

[5] Informacion do algortimo yolo. [En liña]. Dispoñible en: https://appsilon.com/object-detection-yolo-algorithm/

[6] Informacion das redes neuronais convolucionais. [En liña]. Dispoñible en: https://www.juanbarrios.com/redes-neurales-convolucionales/#:~:text=Las%20Redes%20neuronales%20convolucionales%20son,V1)%20de%20un%20cerebro%20biol%C3%B3gico.

[7] Información acerca de psnr. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/PSNR

[8] Información acerca dos tipos de ondas presentes nunha eeg. [En liña]. Dispoñible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3304110/#:~:text=A%20brain%2Dcomputer%20interface%20(BCI,control%20computers%20or%20external%20devices.&text=Finally%2C%20the%20review%20provides%20an,control%20a%20range%20of%20devices

[9] Definición de artefactos. [En liña]. Dispoñible en: https://riunet.upv.es/bitstream/handle/10251/104163/ALAGIA%20-%20Procesamiento%20de%20artefactos%20en%20EEG%20para%20aplicaciones%20de%20comunicaci%C3%B3n%20y%20control.pdf?sequence=1&isAllowed=y

[10] Exemplos de artefactos que se poden encontrar nas eeg. [En liña]. Dispoñible en: https://www.medicine.mcgill.ca/physio/vlab/biomed_signals/eeg_n.htm

[11] Exemplos de sinais eeg. [En liña]. Dispoñible en: https://www.emotiv.com/eeg-guide/

[12] Información sobre os filtros electrónicos. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/Filtro_electr%C3%B3nico#:~:text=Un%20filtro%20el%C3%A9ctrico%20o%20filtro,Activos%20o%20pasivos.

[13] Información sobre os filtros paso banda. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/Filtro_paso_banda

[14] Información sobre mqtt. [En liña]. Dispoñible en: https://descubrearduino.com/mqtt-que-es-como-se-puede-usar-y-como-funciona/

[15] Información acerca de iot. [En liña]. Dispoñible en: https://en.wikipedia.org/wiki/Internet_of_things

[16] Información acerca de patron pub-sub. [En liña]. Dispoñible en: https://www.luisllamas.es/protocolos-de-comunicacion-para-iot/

[17] Informacion python. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/Python

[18] Paxina opencv. [En liña]. Dispoñible en: https://opencv.org/

[19] Dataset utilizado. [En liña]. Dispoñible en: https://cocodataset.org/

[20] Paxina do bci empregado. [En liña]. Dispoñible en: https://shop.openbci.com/products/cyton-biosensing-board-8-channel?variant=38958638542

[21] Tipos de montaxes de electrodos. [En liña]. Dispoñible en: http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S1137-66272009000600006

[22] Explicación dos tipos de electrodos utilizados. [En liña]. Dispoñible en: https://www.biopac.com/knowledge-base/ground-vs-reference-for-eeg-recording/

[23] Ferramenta software do bci utilizado. [En liña]. Dispoñible en: https://docs.openbci.com/docs/06Software/01-OpenBCISoftware/GUIDocs

[24] Información sobre estandar oasis. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/OASIS_(organizaci%C3%B3n)

[25] Información sobre git. [En liña]. Dispoñible en: https://git-scm.com/

[26] Páxina de git da fic. [En liña]. Dispoñible en: https://git.fic.udc.es/users/sign_in

[27] Información acerca de latex. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/LaTeX

[28] Páxina de overleaf. [En liña]. Dispoñible en: https://es.overleaf.com/

[29] Desarrollo iterativo y creciente. [En liña]. Dispoñible en: https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente

[30] Páxina de project libre. [En liña]. Dispoñible en: https://www.projectlibre.com/

[31] Salarios do sector ti. [En liña]. Dispoñible en: https://es.scribd.com/document/288511179/Guia-Salarial-Sector-TI-Galicia-2015-2016

[32] Información acerca dos requisitos dun proxecto. [En liña]. Dispoñible en: https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales

[33] Información acerca do modelo preentrando de yolov3. [En liña]. Dispoñible en: https://pjreddie.com/darknet/yolo/

[34] Páxina de eclipse mosquitto. [En liña]. Dispoñible en: https://mosquitto.org/